Algoritmuselméleti alapok

Hajnal Péter

Bolyai Intézet, TTIK, SZTE, Szeged

2021. ősz

Az algoritmus „fogalma”

Az algoritmus „fogalma”

- Egy feladat algoritmikus megoldása egy eljárás, ami az adatok megkapása után egy jól definiált lépéssort elvégezve kiszámolja a feladatra adandó választ.

Az algoritmus „fogalma”

- Egy feladat algoritmikus megoldása egy eljárás, ami az adatok megkapása után egy jól definiált lépéssort elvégezve kiszámolja a feladatra adandó választ. Minden lépésben egy mindenki számára könnyen elvégezhető elemi utasítást kell végrehajtani.

Az algoritmus „fogalma”

• Egy feladat algoritmikus megoldása egy eljárás, ami az adatok megkapása után egy jól definiált lépéssort elvégezve kiszámolja a feladatra adandó választ. Minden lépésben egy mindenki számára könnyen elvégezhető elemi utasítást kell végrehajtani.

• Azaz egy algoritmuselméleti probléma matematikai modellje a következő „összetevőkből áll”:

Az algoritmus „fogalma”

• Egy feladat algoritmikus megoldása egy eljárás, ami az adatok megkapása után egy jól definiált lépéssort elvégezve kiszámolja a feladatra adandó választ. Minden lépésben egy mindenki számára könnyen elvégezhető elemi utasítást kell végrehajtani.

• Azaz egy algoritmuselméleti probléma matematikai modellje a következő „összetevőkből áll”:

(1) Számítási feladat (adat/eredmény kapcsolat),

Az algoritmus „fogalma”

• Egy feladat algoritmikus megoldása egy eljárás, ami az adatok megkapása után egy jól definiált lépéssort elvégezve kiszámolja a feladatra adandó választ. Minden lépésben egy mindenki számára könnyen elvégezhető elemi utasítást kell végrehajtani.

• Azaz egy algoritmuselméleti probléma matematikai modellje a következő „összetevőkből áll”:

(1) Számítási feladat (adat/eredmény kapcsolat),

(2) Elemi lépések halmaza.

Az algoritmus „fogalma”

• Egy feladat algoritmikus megoldása egy eljárás, ami az adatok megkapása után egy jól definiált lépéssort elvégezve kiszámolja a feladatra adandó választ. Minden lépésben egy mindenki számára könnyen elvégezhető elemi utasítást kell végrehajtani.

• Azaz egy algoritmuselméleti probléma matematikai modellje a következő „összetevőkből áll”:

(1) Számítási feladat (adat/eredmény kapcsolat),

(2) Elemi lépések halmaza.

• Azaz egy algoritmus naív matematikai leírása: Egy leírás, amely egyértelműen magadja, hogy adott input esetén milyen elemi lépések/döntések sorozatával jutunk el az eredményig.

Az algoritmus „fogalma”

• Egy feladat algoritmikus megoldása egy eljárás, ami az adatok megkapása után egy jól definiált lépéssort elvégezve kiszámolja a feladatra adandó választ. Minden lépésben egy mindenki számára könnyen elvégezhető elemi utasítást kell végrehajtani.

• Azaz egy algoritmuselméleti probléma matematikai modellje a következő „összetevőkből áll”:

(1) Számítási feladat (adat/eredmény kapcsolat),

(2) Elemi lépések halmaza.

• Azaz egy algoritmus naív matematikai leírása: Egy leírás, amely egyértelműen magadja, hogy adott input esetén milyen elemi lépések/döntések sorozatával jutunk el az eredményig.

• A naív jelző nem véletlen. Ez nem pontos matematikai definíciója az algoritmusnak.

Az algoritmuselmélet nyelve

Az algoritmuselmélet nyelve

- Az algoritmus fogalmával együtt kialakult egy az algoritmusokkal kapcsolatos nyelvezet is.

Az algoritmuselmélet nyelve

- Az algoritmus fogalmával együtt kialakult egy az algoritmusokkal kapcsolatos nyelvezet is.

- Az adatokat *inputnak* nevezzük.

Az algoritmuselmélet nyelve

- Az algoritmus fogalmával együtt kialakult egy az algoritmusokkal kapcsolatos nyelvezet is.
- Az adatokat *inputnak* nevezzük.
- A közzéadandó eredményt *outputnak* nevezzük.

Az algoritmuselmélet nyelve

• Az algoritmus fogalmával együtt kialakult egy az algoritmusokkal kapcsolatos nyelvezet is.

• Az adatokat *inputnak* nevezzük.

• A közzéadandó eredményt *outputnak* nevezzük.

• Ha adott inputon az algoritmus utasításait követve végezzük az előírt lépéseket, akkor az *algoritmus futásáról* beszélünk az adott inputon.

Általános iskolai algoritmusok

Általános iskolai algoritmusok

- Az alapműveletek elvégezésének szokásos módja is egy-egy algoritmus.

Általános iskolai algoritmusok

- Az alapműveletek elvégezésének szokásos módja is egy-egy algoritmus. Az elemi lépések a számjegyműveletek.

Általános iskolai algoritmusok

- Az alapműveletek elvégezésének szokásos módja is egy-egy algoritmus. Az elemi lépések a számjegyműveletek.

- Az alapszerkesztések algoritmusként is tekinthetők.

Általános iskolai algoritmusok

• Az alapműveletek elvégezésének szokásos módja is egy-egy algoritmus. Az elemi lépések a számjegyműveletek.

• Az alapszerkesztések algoritmusként is tekinthetők. Az elemi lépéseket Euklidesz írta elő.

Általános iskolai algoritmusok

- Az alapműveletek elvégezésének szokásos módja is egy-egy algoritmus. Az elemi lépések a számjegyműveletek.

- Az alapszerkesztések algoritmusként is tekinthetők. Az elemi lépéseket Euklidesz írta elő.

- A prímtényezőkre bontás megtanított módja egy algoritmus.

Általános iskolai algoritmusok

• Az alapműveletek elvégezésének szokásos módja is egy-egy algoritmus. Az elemi lépések a számjegyműveletek.

• Az alapszerkesztések algoritmusként is tekinthetők. Az elemi lépéseket Euklidesz írta elő.

• A prímtényezőkre bontás megtanított módja egy algoritmus. Az elemi lépések az alapműveletek.

Általános iskolai algoritmusok

• Az alapműveletek elvégezésének szokásos módja is egy-egy algoritmus. Az elemi lépések a számjegyműveletek.

• Az alapszerkesztések algoritmusként is tekinthetők. Az elemi lépéseket Euklidesz írta elő.

• A prímtényezőkre bontás megtanított módja egy algoritmus. Az elemi lépések az alapműveletek.

• A legnagyobb közös osztó meghatározására az Euklideszi-algoritmus jól ismertnek kell lenni egy érettségizett számára.

Pontosítás

Pontosítás

- Legyen $\mathcal{I}$ a lehetséges inputok, $\mathcal{O}$ a lehetséges outputok halmaza.

Pontosítás

- Legyen $\mathcal{I}$ a lehetséges inputok, $\mathcal{O}$ a lehetséges outputok halmaza.

Definíció: Algoritmikus/számítási probléma/feladat

Egy algoritmikus/számítási probléma egy $f : \mathcal{I} \to \mathcal{O}$ függvény.

Pontosítás

- Legyen $\mathcal{I}$ a lehetséges inputok, $\mathcal{O}$ a lehetséges outputok halmaza.

Definíció: Algoritmikus/számítási probléma/feladat

Egy algoritmikus/számítási probléma egy $f : \mathcal{I} \to \mathcal{O}$ függvény.

- Azaz a számítási feladat egy függvény kiértékelése.

Pontosítás

- Legyen $\mathcal{I}$ a lehetséges inputok, $\mathcal{O}$ a lehetséges outputok halmaza.

Definíció: Algoritmikus/számítási probléma/feladat

Egy algoritmikus/számítási probléma egy $f : \mathcal{I} \to \mathcal{O}$ függvény.

- Azaz a számítási feladat egy függvény kiértékelése.
- Az elemi lépések (egyelőre) mindig a környezettől a feladattól, annak matematikájától fog függeni.

Történet

Történet

- Az algoritmus fenti leírása nem matematikai definíció. Ezen a ponton nem is adunk matematikai definíciót az algoritmusra.

Történet

• Az algoritmus fenti leírása nem matematikai definíció. Ezen a ponton nem is adunk matematikai definíciót az algoritmusra.

• Megjegyezzük, hogy a matematikus közösség, hosszú vajúdás után az 1930-as években fogadott el egy mind a mai napig használt algoritmus fogalmat.

Történet

• Az algoritmus fenti leírása nem matematikai definíció. Ezen a ponton nem is adunk matematikai definíciót az algoritmusra.

• Megjegyezzük, hogy a matematikus közösség, hosszú vajúdás után az 1930-as években fogadott el egy mind a mai napig használt algoritmus fogalmat.

• Ez jól leírja azt ami a számítógépek használata közben történik.

Történet

• Az algoritmus fenti leírása nem matematikai definíció. Ezen a ponton nem is adunk matematikai definíciót az algoritmusra.

• Megjegyezzük, hogy a matematikus közösség, hosszú vajúdás után az 1930-as években fogadott el egy mind a mai napig használt algoritmus fogalmat.

• Ez jól leírja azt ami a számítógépek használata közben történik.

• Azt is megjegyezzük, ha megjelennének a kvantum-számítógépek, akkor ezt a fogalmat újra kellene értékelnünk.

Történet

• Az algoritmus fenti leírása nem matematikai definíció. Ezen a ponton nem is adunk matematikai definíciót az algoritmusra.

• Megjegyezzük, hogy a matematikus közösség, hosszú vajúdás után az 1930-as években fogadott el egy mind a mai napig használt algoritmus fogalmat.

• Ez jól leírja azt ami a számítógépek használata közben történik.

• Azt is megjegyezzük, ha megjelennének a kvantum-számítógépek, akkor ezt a fogalmat újra kellene értékelnünk. Megjegyzem, hogy az algoritmussal, illetve kvantun algoritmussal kiszámítható problémák halmaza nem változna!

Példa: Számítási problémák

Példa: Számítási problémák

Legtöbbször azonban nem vagyunk ennyire formálisak. A FAKTORIZÁCIÓ például egy számítási probléma. A szóhasználat jelentheti a következők bármelyikét.

Példa: Számítási problémák

Legtöbbször azonban nem vagyunk ennyire formálisak. A FAKTORIZÁCIÓ például egy számítási probléma. A szóhasználat jelentheti a következők bármelyikét.

Példa (FAKTORIZÁCIÓ I)

Input: egy pozitív egész szám. Output: prím osztóinak listája a megfelelő multiplicitásokkal.

Példa: Számítási problémák

Legtöbbször azonban nem vagyunk ennyire formálisak. A FAKTORIZÁCIÓ például egy számítási probléma. A szóhasználat jelentheti a következők bármelyikét.

Példa (FAKTORIZÁCIÓ I)

Input: egy pozitív egész szám. Output: prím osztóinak listája a megfelelő multiplicitásokkal.

Példa (FAKTORIZÁCIÓ II)

Input: egy pozitív egész szám. Output: egy prím osztója.

Példa (folytatás): Számítási problémák

Példa (folytatás): Számítási problémák

Példa (FAKTORIZÁCIÓ III)

Input: egy pozitív egész szám. Output: legkisebb prím osztója.

Példa (folytatás): Számítási problémák

Példa (FAKTORIZÁCIÓ III)

Input: egy pozitív egész szám. Output: legkisebb prím osztója.

Példa (FAKTORIZÁCIÓ IV)

Input: egy pozitív egész szám és egy t érték. Döntsük el van-e 2 és t közötti osztó.

Példa (folytatás): Számítási problémák

Példa (FAKTORIZÁCIÓ III)

Input: egy pozitív egész szám. Output: legkisebb prím osztója.

Példa (FAKTORIZÁCIÓ IV)

Input: egy pozitív egész szám és egy t érték. Döntsük el van-e 2 és t közötti osztó.

Bármelyik megoldása átalakítható (alap programozási technikák, például iteráció/rekurzió, bináris keresés ismeretével) a többi megoldásává.

Kódolás, az input mérete

Kódolás, az input mérete

- Az inputok és outputok leírásában/leírtak értelmezésben is meg kell egyezniük a számítást követőknek.

Kódolás, az input mérete

• Az inputok és outputok leírásában/leírtak értelmezésben is meg kell egyezniük a számítást követőknek.

• Gyakorlatban egy géppel kell „közölnünk” az adatokat/inputot, az output kiszámolása után pedig a gép által adott végeredményt kell „értelmeznünk”.

Kódolás, az input mérete

• Az inputok és outputok leírásában/leírtak értelmezésben is meg kell egyezniük a számítást követőknek.

• Gyakorlatban egy géppel kell „közölnünk” az adatokat/inputot, az output kiszámolása után pedig a gép által adott végeredményt kell „értelmeznünk”.

• Hogy az adatok, hogyan kódoltak az kérdéses. Néha nem is olyan fontos, néha tisztázni kell az alapokat.

Kódolás, az input mérete

- Az inputok és outputok leírásában/leírtak értelmezésben is meg kell egyezniük a számítást követőknek.

- Gyakorlatban egy géppel kell „közölnünk” az adatokat/inputot, az output kiszámolása után pedig a gép által adott végeredményt kell „értelmeznünk”.

- Hogy az adatok, hogyan kódoltak az kérdéses. Néha nem is olyan fontos, néha tisztázni kell az alapokat.

- Az input kódolása elvezet az input méretének fogalmához.

Kódolás: Példa

Kódolás: Példa

Példa

Adott n valós szám $(x_1, \ldots, x_n)$, határozzuk meg rendezett sorrendjüket.

Kódolás: Példa

Példa

Adott n valós szám $(x_1, \ldots, x_n)$, határozzuk meg rendezett sorrendjüket.

- A valós számok halmaza kontinuum számosságú.

Kódolás: Példa

Példa

Adott n valós szám $(x_1, \ldots, x_n)$, határozzuk meg rendezett sorrendjüket.

- A valós számok halmaza kontinuum számosságú.
- Egy valódi számítógép véges 0-1 sorozatokat képes tárolni. A kódok egy megszámlálhatóan végtelen halmazt alkotnak.

Kódolás: Példa

Példa

Adott n valós szám $(x_1, \ldots, x_n)$, határozzuk meg rendezett sorrendjüket.

- A valós számok halmaza kontinuum számosságú.
- Egy valódi számítógép véges 0-1 sorozatokat képes tárolni. A kódok egy megszámlálhatóan végtelen halmazt alkotnak.
- Ez „túl kicsi” az összes valós szám kezelésére. Ennek ellenére ez egy fontos feladat.

Kódolás: Példa, 1. folytatás

Kódolás: Példa, 1. folytatás

- Egy lehetőség, hogy feltesszük adatain mérések eredményei, racionális (egész) számok.

Kódolás: Példa, 1. folytatás

- Egy lehetőség, hogy feltesszük adatain mérések eredményei, racionális (egész) számok.
- Az input mérete a leíráshoz szükséges számjegyek/karakterek száma.

Kódolás: Példa, 2. folytatás

Kódolás: Példa, 2. folytatás

- Az elemi lépésekből egyre szorítkozunk: „két szám összehasonlítása”.

Kódolás: Példa, 2. folytatás

- Az elemi lépésekből egyre szorítkozunk: „két szám összehasonlítása”.
- Ekkor feltehetjük, hogy az x_i-k valóban valós számok.

Kódolás: Példa, 2. folytatás

- Az elemi lépésekből egyre szorítkozunk: „két szám összehasonlítása”.
- Ekkor feltehetjük, hogy az x_i-k valóban valós számok. Mi rámutatunk két számra és az algoritmus egy elemi lépéssel (egy költséggel) megadja a választ

Kódolás: Példa, 2. folytatás

- Az elemi lépésekből egyre szorítkozunk: „két szám összehasonlítása”.
- Ekkor feltehetjük, hogy az x_i-k valóban valós számok. Mi rámutatunk két számra és az algoritmus egy elemi lépéssel (egy költséggel) megadja a választ Azt mondjuk az x_i-ket pontos, valós aritmetikával képzeljük el.

Kódolás: Példa, 2. folytatás

- Az elemi lépésekből egyre szorítkozunk: „két szám összehasonlítása”.
- Ekkor feltehetjük, hogy az x_i-k valóban valós számok. Mi rámutatunk két számra és az algoritmus egy elemi lépéssel (egy költséggel) megadja a választ Azt mondjuk az x_i-ket pontos, valós aritmetikával képzeljük el. **Ekkor az input mérete n.**

Kódolás: Példa, 2. folytatás

- Az elemi lépésekből egyre szorítkozunk: „két szám összehasonlítása”.
- Ekkor feltehetjük, hogy az x_i-k valóban valós számok. Mi rámutatunk két számra és az algoritmus egy elemi lépéssel (egy költséggel) megadja a választ Azt mondjuk az x_i-ket pontos, valós aritmetikával képzeljük el. **Ekkor az input mérete n.**
- Azaz elképezelünk egy olyan számítógépet, amiben egy bit tárolására alkalmas memória egység helyett egy doboz szerepel képzeletünkben. Ezekbe a dobozokba valós számokat rakhatunk.

Kódolás: Példa, 2. folytatás

- Az elemi lépésekből egyre szorítkozunk: „két szám összehasonlítása”.
- Ekkor feltehetjük, hogy az x_i-k valóban valós számok. Mi rámutatunk két számra és az algoritmus egy elemi lépéssel (egy költséggel) megadja a választ Azt mondjuk az x_i-ket pontos, valós aritmetikával képzeljük el. **Ekkor az input mérete n.**
- Azaz elképezelünk egy olyan számítógépet, amiben egy bit tárolására alkalmas memória egység helyett egy doboz szerepel képzeletünkben. Ezekbe a dobozokba valós számokat rakhatunk. A dobozok „tartalmán” végzünk műveleteket.

Kódolás: Példa, 2. folytatás

• Az elemi lépésekből egyre szorítkozunk: „két szám összehasonlítása”.

• Ekkor feltehetjük, hogy az x_i-k valóban valós számok. Mi rámutatunk két számra és az algoritmus egy elemi lépéssel (egy költséggel) megadja a választ Azt mondjuk az x_i-ket pontos, valós aritmetikával képzeljük el. **Ekkor az input mérete n.**

• Azaz elképezelünk egy olyan számítógépet, amiben egy bit tárolására alkalmas memória egység helyett egy doboz szerepel képzeletünkben. Ezekbe a dobozokba valós számokat rakhatunk. A dobozok „tartalmán” végzünk műveleteket.

• Ez a való életben nem lehetséges, habár írhatunk programot, amiben két valósnak deklarált számot összeszorozhatunk. A program futása során egy memória területen egy kerekített érték tárolódik és a szorzás után a szorzat a kerekített értékek szorzatának kerekített értéke lesz.

Példa

Példa

Egy teljesen más probléma a következő:

Példa

Egy teljesen más probléma a következő:

Példa

Adott n darab k bites szám, határozzuk meg rendezett sorrendjüket.

Példa

Egy teljesen más probléma a következő:

Példa

Adott n darab k bites szám, határozzuk meg rendezett sorrendjüket.

Itt a szövegezésből nyilvánvaló, hogy az input mérete $n \cdot k$ bit és bit műveletekben kell gondolkoznunk mint elemi műveletek.

Példa

Példa

Példa

Adott két egész szám, számoljuk ki összegüket/különbségüket/szorzatukat.

Példa

Példa

Adott két egész szám, számoljuk ki összegüket/különbségüket/szorzatukat.

- Az elemi lépés a számjegyek közötti elemi műveletek.

Példa

Példa

Adott két egész szám, számoljuk ki összegüket/különbségüket/szorzatukat.

- Az elemi lépés a számjegyek közötti elemi műveletek.
- Például a szorzótábla kétjegyű végeredményeinek továbbviteli jegyre/utolsó számjegyre való szétszedése.

Példa

Példa

Adott két egész szám, számoljuk ki összegüket/különbségüket/szorzatukat.

- Az elemi lépés a számjegyek közötti elemi műveletek.
- Például a szorzótábla kétjegyű végeredményeinek továbbviteli jegyre/utolsó számjegyre való szétszedése.
- Az input mérete „karakter számolás”.

Példa

Példa

Példa

Adott két természetes szám, számoljuk ki legnagyobb közös osztójukat.

Példa

Példa

Adott két természetes szám, számoljuk ki legnagyobb közös osztójukat.

- Az elemi lépések az egészek közötti összehasonlítás/összeg/különbség képzés esetleg szorzás, maradékos osztás.

Példa

Példa

Adott két természetes szám, számoljuk ki legnagyobb közös osztójukat.

- Az elemi lépések az egészek közötti összehasonlítás/összeg/különbség képzés esetleg szorzás, maradékos osztás.

- Azaz most két szám összege/különbsége egyetlen elemi lépésnek számít.

Példa

Példa

Adott két természetes szám, számoljuk ki legnagyobb közös osztójukat.

- Az elemi lépések az egészek közötti összehasonlítás/összeg/különbség képzés esetleg szorzás, maradékos osztás.

- Azaz most két szám összege/különbsége egyetlen elemi lépésnek számít.

- Az input mérete „karakter számolás”.

Nézőpontok viszonya

Nézőpontok viszonya

- Ha valakit zavar, hogy most egy összeadás (megtanult algoritmussal elvégezhető probbléma) egy elemi lépés (költsége 1), akkor a következőt teheti.

Nézőpontok viszonya

• Ha valakit zavar, hogy most egy összeadás (megtanult algoritmussal elvégezhető probbléma) egy elemi lépés (költsége 1), akkor a következőt teheti.

• A magasabb szintű algoritmus leírásban az $z \leftarrow x + y$ összeadásokat (mint elemi lépéseket) mint egyetlen lépést helyettesítse a korábbi példa alapműveleti algoritmusával.

Példa: Mátrix szorzás

Példa: Mátrix szorzás

Példa

Adott $A, B \in \mathbb{R}^{n \times n}$ két mátrix. Számítsuk ki szorzatukat.

Példa: Mátrix szorzás

Példa

Adott $A, B \in \mathbb{R}^{n \times n}$ két mátrix. Számítsuk ki szorzatukat.

- Ismét „pontos valós aritmetikát sugallunk”.

Példa: Mátrix szorzás

Példa

Adott $A, B \in \mathbb{R}^{n \times n}$ két mátrix. Számítsuk ki szorzatukat.

- Ismét „pontos valós aritmetikát sugallunk”.
- Nézhetjük, hogy a mátrix szorzás definíciója szerint hány szorzás, összegzés (elemi lépés) történik.

Példa: Mátrix szorzás

Példa

Adott $A, B \in \mathbb{R}^{n \times n}$ két mátrix. Számítsuk ki szorzatukat.

- Ismét „pontos valós aritmetikát sugallunk”.
- Nézhetjük, hogy a mátrix szorzás definíciója szerint hány szorzás, összegzés (elemi lépés) történik.
- Ez az analízis nem a valóságtól elrugaszkodott, habár egy elméleti géppel dolgozik. Nagyon fontos a mátrix szorzás fenti típusú analízise.

Példa: Mátrix szorzás

Példa

Adott $A, B \in \mathbb{R}^{n \times n}$ két mátrix. Számítsuk ki szorzatukat.

- Ismét „pontos valós aritmetikát sugallunk”.

- Nézhetjük, hogy a mátrix szorzás definíciója szerint hány szorzás, összegzés (elemi lépés) történik.

- Ez az analízis nem a valóságtól elrugaszkodott, habár egy elméleti géppel dolgozik. Nagyon fontos a mátrix szorzás fenti típusú analízise.

- A fenti példában az input mérete is fontos. Két mátrix adott, de az input méretét 2-nek venni csalás lenne. A természetes megállapodás, hogy valós számok „seregének” gondoljuk az inputot, $2n^2$ számra gondolunk A, B helyett. Az sem nagy csalás, ha az n paraméterrel jelöljük mekkora feladatról van szó.

Példa: Háromszögek számlálása

Példa: Háromszögek számlálása

Példa

Adott egy egyszerű gráf határozzuk meg hány háromszög (három hosszú kör) van benne.

Példa: Háromszögek számlálása

Példa

Adott egy egyszerű gráf határozzuk meg hány háromszög (három hosszú kör) van benne.

• Hogyan adott egy n csúcsú egyszerű gráf? Több (mondhatjuk sok) lehetőség van. Csak kettőt emelünk ki.

Kódolás: Gráfok kódolása I

Kódolás: Gráfok kódolása I

- Első megadási módszer lehet a szomszédsági mátrix (egy $n \times n$ bit-mátrix) megadása.

Kódolás: Gráfok kódolása I

• Első megadási módszer lehet a szomszédsági mátrix (egy $n \times n$ bit-mátrix) megadása.

• Egy kicsit pazarló módszer, hiszen a mátrix főátlóján nullák szerepelnek és szimmetrikus. Azaz igazából $\binom{n}{2}$ független bit a kódolás. Ennek ellenére a mátrix struktúra, algebra gyakran nagyon hasznos.

Kódolás: Gráfok kódolása II

Kódolás: Gráfok kódolása II

- Egy másik lehetőség, hogy vesszük a csúcsok egy listáját.

Kódolás: Gráfok kódolása II

- Egy másik lehetőség, hogy vesszük a csúcsok egy listáját.
- A lista minden v eleméhez hozzátartozik a v csúcs szomszédjainak egy listája (egy s szomszéd igazából a vs élt reprezentálja).

Kódolás: Gráfok kódolása II

- Egy másik lehetőség, hogy vesszük a csúcsok egy listáját.
- A lista minden v eleméhez hozzátartozik a v csúcs szomszédjainak egy listája (egy s szomszéd igazából a vs élt reprezentálja).
- A szomszédok listáiban minden szomszéd tartalmazza a `következő-szomszéd` infot, ami erre a szomszédra mutat

Kódolás: Gráfok kódolása II

- Egy másik lehetőség, hogy vesszük a csúcsok egy listáját.
- A lista minden v eleméhez hozzátartozik a v csúcs szomszédjainak egy listája (egy s szomszéd igazából a vs élt reprezentálja).
- A szomszédok listáiban minden szomszéd tartalmazza a `következő-szomszéd` infot, ami erre a szomszédra mutat (amennyiben ez a szomszéd létezik, amennyiben az utolsó szomszédról beszélünk mondjuk egy speciális NIL értéket vesz fel).

Kódolás: Gráfok kódolása II

- Egy másik lehetőség, hogy vesszük a csúcsok egy listáját.
- A lista minden v eleméhez hozzátartozik a v csúcs szomszédjainak egy listája (egy s szomszéd igazából a vs élt reprezentálja).
- A szomszédok listáiban minden szomszéd tartalmazza a `következő-szomszéd` infot, ami erre a szomszédra mutat (amennyiben ez a szomszéd létezik, amennyiben az utolsó szomszédról beszélünk mondjuk egy speciális NIL értéket vesz fel).
- A teljes csúcs-listájában minden csúcshoz tartozik egy `első-szomszéd` info, ami lehet NIL is ha a csúcs izolált.

Kódolás: Gráfok kódolása II

- Egy másik lehetőség, hogy vesszük a csúcsok egy listáját.
- A lista minden v eleméhez hozzátartozik a v csúcs szomszédjainak egy listája (egy s szomszéd igazából a vs élt reprezentálja).
- A szomszédok listáiban minden szomszéd tartalmazza a `következő-szomszéd` infot, ami erre a szomszédra mutat (amennyiben ez a szomszéd létezik, amennyiben az utolsó szomszédról beszélünk mondjuk egy speciális NIL értéket vesz fel).
- A teljes csúcs-listájában minden csúcshoz tartozik egy `első-szomszéd` info, ami lehet NIL is ha a csúcs izolált.
- Persze ott van a `következő-csúcs` info is, ami a teljes csúcs-lista utolsó csúcsánál NIL értékű. Persze lehetnek további információk is a struktúrában.

Kódolás: Gráfok kódolása II

- Egy másik lehetőség, hogy vesszük a csúcsok egy listáját.
- A lista minden v eleméhez hozzátartozik a v csúcs szomszédjainak egy listája (egy s szomszéd igazából a vs élt reprezentálja).
- A szomszédok listáiban minden szomszéd tartalmazza a `következő-szomszéd` infot, ami erre a szomszédra mutat (amennyiben ez a szomszéd létezik, amennyiben az utolsó szomszédról beszélünk mondjuk egy speciális NIL értéket vesz fel).
- A teljes csúcs-listájában minden csúcshoz tartozik egy `első-szomszéd` info, ami lehet NIL is ha a csúcs izolált.
- Persze ott van a `következő-csúcs` info is, ami a teljes csúcs-lista utolsó csúcsánál NIL értékű. Persze lehetnek további információk is a struktúrában.
- Például egy élsúlyozott gráf esetén a v csúcs s szomszédjánál szerepelhet a vs él súlya (`súly`), ami mondjuk egy pozitív egész.

Kódolás: Gráfok kódolása II

- Egy másik lehetőség, hogy vesszük a csúcsok egy listáját.
- A lista minden v eleméhez hozzátartozik a v csúcs szomszédjainak egy listája (egy s szomszéd igazából a vs élt reprezentálja).
- A szomszédok listáiban minden szomszéd tartalmazza a `következő-szomszéd` infot, ami erre a szomszédra mutat (amennyiben ez a szomszéd létezik, amennyiben az utolsó szomszédról beszélünk mondjuk egy speciális NIL értéket vesz fel).
- A teljes csúcs-listájában minden csúcshoz tartozik egy `első-szomszéd` info, ami lehet NIL is ha a csúcs izolált.
- Persze ott van a `következő-csúcs` info is, ami a teljes csúcs-lista utolsó csúcsánál NIL értékű. Persze lehetnek további információk is a struktúrában.
- Például egy élsúlyozott gráf esetén a v csúcs s szomszédjánál szerepelhet a vs él súlya (`súly`), ami mondjuk egy pozitív egész.
- Így a gráf összetett adatok bonyolultan kapcsolt összessége.

Összegezve

Összegezve

- A két ábrázolás más és más elemi műveleteket sugall és ugyanannak a magas szinten leírt algoritmusnak a megvalósítása, elemzése teljesen más kérdéseket vet fel.

Összegezve

- A két ábrázolás más és más elemi műveleteket sugall és ugyanannak a magas szinten leírt algoritmusnak a megvalósítása, elemzése teljesen más kérdéseket vet fel.

- Lehet, hogy a hallgatót a fenti példák összezavarják.

Összegezve

- A két ábrázolás más és más elemi műveleteket sugall és ugyanannak a magas szinten leírt algoritmusnak a megvalósítása, elemzése teljesen más kérdéseket vet fel.

- Lehet, hogy a hallgatót a fenti példák összezavarják.

- Nem ez volt a cél.

Összegezve

• A két ábrázolás más és más elemi műveleteket sugall és ugyanannak a magas szinten leírt algoritmusnak a megvalósítása, elemzése teljesen más kérdéseket vet fel.

• Lehet, hogy a hallgatót a fenti példák összezavarják.

• Nem ez volt a cél. A cél, hogy egy algoritmikus problémát jól át kell gondolni.

Összegezve

• A két ábrázolás más és más elemi műveleteket sugall és ugyanannak a magas szinten leírt algoritmusnak a megvalósítása, elemzése teljesen más kérdéseket vet fel.

• Lehet, hogy a hallgatót a fenti példák összezavarják.

• Nem ez volt a cél. A cél, hogy egy algoritmikus problémát jól át kell gondolni. Ha többen tárgyaljuk, akkor sokat kell kérdezni, tisztázni, hogy egy matematikailag megalapozott tárgyalás alakuljon ki.

Szünet

Legrosszabb eset analízis

Legrosszabb eset analízis

- Adott ω inputon futtatva az algoritmust meg kell számolnunk (néha megbecsülnünk) hány elemi lépés vezet el az outputhoz.

Legrosszabb eset analízis

- Adott ω inputon futtatva az algoritmust meg kell számolnunk (néha megbecsülnünk) hány elemi lépés vezet el az outputhoz.

- Legyen $\mathcal{A}$ egy algoritmus, ω egy input.

Legrosszabb eset analízis

- Adott ω inputon futtatva az algoritmust meg kell számolnunk (néha megbecsülnünk) hány elemi lépés vezet el az outputhoz.

- Legyen $\mathcal{A}$ egy algoritmus, ω egy input.

- $t_{\mathcal{A}}(\omega)$ az elemi lépések száma ami szükséges, hogy ω inputon futtatva $\mathcal{A}$-t megkapjuk az outputot.

Legrosszabb eset analízis

• Adott ω inputon futtatva az algoritmust meg kell számolnunk (néha megbecsülnünk) hány elemi lépés vezet el az outputhoz.

• Legyen $\mathcal{A}$ egy algoritmus, ω egy input.

• $t_{\mathcal{A}}(\omega)$ az elemi lépések száma ami szükséges, hogy ω inputon futtatva $\mathcal{A}$-t megkapjuk az outputot.

• Használtuk az input méretének fogalmát is. Azaz $\mathcal{I} = \cup_{s=0}^{\infty} \mathcal{I}_s$, ahol $\mathcal{I}_s$ az s méretű inputok halmaza.

Algoritmusok idő analízise

Definíció

$$t_{\mathcal{A}}(n) = \max\{t_{\mathcal{A}}(\omega) : \omega \in \mathcal{I}_n\}.$$

Algoritmusok idő analízise

Definíció

$$t_{\mathcal{A}}(n) = \max\{t_{\mathcal{A}}(\omega) : \omega \in \mathcal{I}_n\}.$$

• A fenti definíció nagyon fontos. Amikor az ebben definiált függvényt vizsgáljuk — például egy jó felső becslést adunk rá — akkor azt mondjuk a *legrosszabb eset analízist* végezzük el.

Algoritmusok idő analízise

Definíció

$$t_{\mathcal{A}}(n) = \max\{t_{\mathcal{A}}(\omega) : \omega \in \mathcal{I}_n\}.$$

• A fenti definíció nagyon fontos. Amikor az ebben definiált függvényt vizsgáljuk — például egy jó felső becslést adunk rá — akkor azt mondjuk a *legrosszabb eset analízist* végezzük el.

• Valóban, ha az $\mathcal{I}_n$ inputokat vizsgáljuk, akkor a maximum vétele a legtöbb elemi lépés számot veszi amit ezen inputok között végre kell hajtanunk az output meghatározásához.

Algoritmusok idő analízise

Definíció

$$t_{\mathcal{A}}(n) = \max\{t_{\mathcal{A}}(\omega) : \omega \in \mathcal{I}_n\}.$$

• A fenti definíció nagyon fontos. Amikor az ebben definiált függvényt vizsgáljuk — például egy jó felső becslést adunk rá — akkor azt mondjuk a *legrosszabb eset analízist* végezzük el.

• Valóban, ha az $\mathcal{I}_n$ inputokat vizsgáljuk, akkor a maximum vétele a legtöbb elemi lépés számot veszi amit ezen inputok között végre kell hajtanunk az output meghatározásához.

• Azaz egy felső becslés az összes adott nagyságú inputon való futás esetén. Azaz egy olyan „garancia”, ami az input hosszának függvénye.

Példa: Háromszög számlálás I

Példa: Háromszög számlálás I

Algoritmus

Példa: Háromszög számlálás I

Algoritmus

0) Legyen $\Delta = 0$

Példa: Háromszög számlálás I

Algoritmus

0) Legyen $\Delta = 0$

1) Vegyük sorra az $e = uv$ éleket.

Példa: Háromszög számlálás I

Algoritmus

0) Legyen $\Delta = 0$

1) Vegyük sorra az $e = uv$ éleket.

1_e) Minden $e = uv$ élnél nézzük meg, hogy u és v-nek mik a közös szomszédai. Minden megtalált közös szomszédra $\Delta \leftarrow \Delta + 1$.

Példa: Háromszög számlálás I

Algoritmus

0) Legyen $\Delta = 0$

1) Vegyük sorra az $e = uv$ éleket.

1_e) Minden $e = uv$ élnél nézzük meg, hogy u és v-nek mik a közös szomszédai. Minden megtalált közös szomszédra $\Delta \leftarrow \Delta + 1$.

2) Írjuk ki $\Delta/3$-ot.

Analízis: Háromszög számlálás I

Analízis: Háromszög számlálás I

- A listás reprezentációval adott G gráf az input.

Analízis: Háromszög számlálás I

- A listás reprezentációval adott G gráf az input.
- 1_e-t $|E|$-szer hajtjuk végre.

Analízis: Háromszög számlálás I

- A listás reprezentációval adott G gráf az input.
- 1_e-t $|E|$-szer hajtjuk végre.
- 1_e időigénye legfeljebb D^2.

Analízis: Háromszög számlálás I

- A listás reprezentációval adott G gráf az input.
- 1_e-t $|E|$-szer hajtjuk végre.
- 1_e időigénye legfeljebb D^2.(?)

Analízis: Háromszög számlálás I

- A listás reprezentációval adott G gráf az input.
- 1_e-t $|E|$-szer hajtjuk végre.
- 1_e időigénye legfeljebb D^2.(?)
- A teljes futási legfeljebb

$$D^2 \cdot |E|.$$

Példa: Háromszög számlálás II

Példa: Háromszög számlálás II

Algoritmus

Példa: Háromszög számlálás II

Algoritmus

1) Számoljuk ki A_G^3-t.

Példa: Háromszög számlálás II

Algoritmus

1) Számoljuk ki A_G^3-t.

2) Írjuk ki $TrA_G^3/6$ értékét.

Analizis: Háromszög számlálás II

Analizis: Háromszög számlálás II

- A szomszédsági mátrixával adott a G gráf, az input.

Analizis: Háromszög számlálás II

- A szomszédsági mátrixával adott a G gráf, az input.
- Egy mátrixszorzáshoz kell(?) $|V|^3$ szorzás és $|V|^3$ összedás.

Analizis: Háromszög számlálás II

- A szomszédsági mátrixával adott a G gráf, az input.
- Egy mátrixszorzáshoz kell(?) $|V|^3$ szorzás és $|V|^3$ összedás.
- A^3 kiszámolásához kell $4|V|^3$ alapmúvelet.

Analizis: Háromszög számlálás II

- A szomszédsági mátrixával adott a G gráf, az input.
- Egy mátrixszorzáshoz kell(?) $|V|^3$ szorzás és $|V|^3$ összedás.
- A^3 kiszámolásához kell $4|V|^3$ alapmúvelet.
- A teljes futási legfeljebb

$$4|V|^3 + |V|.$$

Egy fontos megjegyzés

Egy fontos megjegyzés

- Egy pontos analízis nagyon hosszú, összetett, áttekinthetetlen formulákhoz vezet.

Egy fontos megjegyzés

- Egy pontos analízis nagyon hosszú, összetett, áttekinthetetlen formulákhoz vezet.
- Másrészt az analízis az algoritmus futásához szükséges időt írja le.

Egy fontos megjegyzés

- Egy pontos analízis nagyon hosszú, összetett, áttekinthetetlen formulákhoz vezet.
- Másrészt az analízis az algoritmus futásához szükséges időt írja le.
- De milyen mértékegységben?

Egy fontos megjegyzés

• Egy pontos analízis nagyon hosszú, összetett, áttekinthetetlen formulákhoz vezet.

• Másrészt az analízis az algoritmus futásához szükséges időt írja le.

• De milyen mértékegységben? Ha másodpercben mérnénk a futást, akkor az elemi lépések száma mellett egy szorzó szerepel, ami az elemi lépésekhez szükséges valódi idő.

Egy fontos megjegyzés

• Egy pontos analízis nagyon hosszú, összetett, áttekinthetetlen formulákhoz vezet.

• Másrészt az analízis az algoritmus futásához szükséges időt írja le.

• De milyen mértékegységben? Ha másodpercben mérnénk a futást, akkor az elemi lépések száma mellett egy szorzó szerepel, ami az elemi lépésekhez szükséges valódi idő.

• Ez a szorzó függ a gépünk hardware-étől/paramétereitől. Egy öt éves gép lecserélese egy maira jelentősen befolyásolja ezt a szorzót.

Egy fontos megjegyzés

- Egy pontos analízis nagyon hosszú, összetett, áttekinthetetlen formulákhoz vezet.
- Másrészt az analízis az algoritmus futásához szükséges időt írja le.
- De milyen mértékegységben? Ha másodpercben mérnénk a futást, akkor az elemi lépések száma mellett egy szorzó szerepel, ami az elemi lépésekhez szükséges valódi idő.
- Ez a szorzó függ a gépünk hardware-étől/paramétereitől. Egy öt éves gép lecserélese egy maira jelentősen befolyásolja ezt a szorzót.
- Így az analízis első fázisában a konstans szorzók nem lényegesek.

Egy fontos megjegyzés

- Egy pontos analízis nagyon hosszú, összetett, áttekinthetetlen formulákhoz vezet.
- Másrészt az analízis az algoritmus futásához szükséges időt írja le.
- De milyen mértékegységben? Ha másodpercben mérnénk a futást, akkor az elemi lépések száma mellett egy szorzó szerepel, ami az elemi lépésekhez szükséges valódi idő.
- Ez a szorzó függ a gépünk hardware-étől/paramétereitől. Egy öt éves gép lecserélese egy maira jelentősen befolyásolja ezt a szorzót.
- Így az analízis első fázisában a konstans szorzók nem lényegesek.
- Általában elhanyagoljuk. Hogy ezt megtehesük/formalizáljuk néhány fontos matematikai jelölésre van szükség.

Nagy ordó jelölés

Nagy ordó jelölés

Definíció

Legyen $t, f : \mathbb{N} \to \mathbb{R}$. Ekkor $t = \mathcal{O}(f)$ azt jelenti, hogy alkalmas $c > 0$ konstansra és n_0 küszöbértékre fennáll, hogy $n > n_0$ esetén

$$|t(n)| \leq cf(n)$$

Nagy ordó jelölés

Definíció

Legyen $t, f : \mathbb{N} \to \mathbb{R}$. Ekkor $t = \mathcal{O}(f)$ azt jelenti, hogy alkalmas $c > 0$ konstansra és n_0 küszöbértékre fennáll, hogy $n > n_0$ esetén

$$|t(n)| \leq cf(n)$$

- $t(n)$ az analízált idő, ez általában egy nemnegatív értékeket felvevő függvény.

Nagy ordó jelölés

Definíció

Legyen $t, f : \mathbb{N} \to \mathbb{R}$. Ekkor $t = \mathcal{O}(f)$ azt jelenti, hogy alkalmas $c > 0$ konstansra és n_0 küszöbértékre fennáll, hogy $n > n_0$ esetén

$$|t(n)| \leq cf(n)$$

- $t(n)$ az analízált idő, ez általában egy nemnegatív értékeket felvevő függvény.

- $f(n)$ egy „egyszerű” függvény, mint n, n^2, n^{10}, $n \log n$, 2^n, n^n, 2^{n^2}.

Nagy ordó jelölés

Definíció

Legyen $t, f : \mathbb{N} \to \mathbb{R}$. Ekkor $t = \mathcal{O}(f)$ azt jelenti, hogy alkalmas $c > 0$ konstansra és n_0 küszöbértékre fennáll, hogy $n > n_0$ esetén

$$|t(n)| \leq cf(n)$$

- $t(n)$ az analízált idő, ez általában egy nemnegatív értékeket felvevő függvény.

- $f(n)$ egy „egyszerű” függvény, mint n, n^2, n^{10}, $n \log n$, 2^n, n^n, 2^{n^2}.

Nagy ordó jelölés

Definíció

Legyen $t, f : \mathbb{N} \to \mathbb{R}$. Ekkor $t = \mathcal{O}(f)$ azt jelenti, hogy alkalmas $c > 0$ konstansra és n_0 küszöbértékre fennáll, hogy $n > n_0$ esetén

$$|t(n)| \leq cf(n)$$

- $t(n)$ az analízált idő, ez általában egy nemnegatív értékeket felvevő függvény.

- $f(n)$ egy „egyszerű” függvény, mint n, n^2, n^{10}, $n \log n$, 2^n, n^n, 2^{n^2}.

- Álljunk meg. Az $f(n) = n \log n$ példában nem írtam oda a logaritmus alapját. Melyik alapra gondoltam? Számít a pontos érték?

Példa

Példa

Példa

$$18\binom{n}{6} + 127n^4 \log n + \log^{15}(n^{124}) \cdot n + 144 = \mathcal{O}(n^6).$$

Példa

Példa

$$18\binom{n}{6} + 127n^4 \log n + \log^{15}(n^{124}) \cdot n + 144 = \mathcal{O}(n^6).$$

- $n \log n$ tényleg $\mathbb{N}$-en értelmezett? A küszöbérték szereplése miatt nem nagyon zavar, hogy 0-ban nem értelmezett.

Példa

Példa

$$18\binom{n}{6} + 127n^4 \log n + \log^{15}(n^{124}) \cdot n + 144 = \mathcal{O}(n^6).$$

- $n \log n$ tényleg $\mathbb{N}$-en értelmezett? A küszöbérték szereplése miatt nem nagyon zavar, hogy 0-ban nem értelmezett.

- Mondhatnánk, hogy kicsire nem adunk, de ez nem matematikusi hozzáállás.

Példa

Példa

$$18\binom{n}{6} + 127n^4 \log n + \log^{15}(n^{124}) \cdot n + 144 = \mathcal{O}(n^6).$$

• $n \log n$ tényleg $\mathbb{N}$-en értelmezett? A küszöbérték szereplése miatt nem nagyon zavar, hogy 0-ban nem értelmezett.

• Mondhatnánk, hogy kicsire nem adunk, de ez nem matematikusi hozzáállás.

• Az átláthatóság és pontosság között kell egyensúlyozni egy olyan területen, ahol nagyon sok NEM matematikus dolgozik.

Példa

Példa

$$18\binom{n}{6} + 127n^4 \log n + \log^{15}(n^{124}) \cdot n + 144 = \mathcal{O}(n^6).$$

• $n \log n$ tényleg $\mathbb{N}$-en értelmezett? A küszöbérték szereplése miatt nem nagyon zavar, hogy 0-ban nem értelmezett.

• Mondhatnánk, hogy kicsire nem adunk, de ez nem matematikusi hozzáállás.

• Az átláthatóság és pontosság között kell egyensúlyozni egy olyan területen, ahol nagyon sok NEM matematikus dolgozik.

• Ezért sokszor a matematikus szemüveg sok végiggondolnivalót/pontosító megjegyzést hagy az olvasó számára. Kérdezzünk, konzultáljunk ...

Omega jelölés

Omega jelölés

- Természetesen $n^2 = \mathcal{O}(2^n)$.

Omega jelölés

- Természetesen $n^2 = \mathcal{O}(2^n)$.

- De egy négyzetes függvényt exponenciálissal becsülve nagyon hanyagok vagyunk. További jelölések segítik azt, hogy a pontos nagyságrendet kiemeljük.

Omega jelölés

- Természetesen $n^2 = \mathcal{O}(2^n)$.

- De egy négyzetes függvényt exponenciálissal becsülve nagyon hanyagok vagyunk. További jelölések segítik azt, hogy a pontos nagyságrendet kiemeljük.

Definíció

Legyen $t, f : \mathbb{N} \to \mathbb{R}$. Ekkor $t = \Omega(f)$ azt jelenti, hogy alkalmas $c > 0$ konstansra és n_0 küszöbértékre fennáll, hogy $n > n_0$ esetén

$$cf(n) \leq t(n).$$

Omega jelölés

- Természetesen $n^2 = \mathcal{O}(2^n)$.

- De egy négyzetes függvényt exponenciálissal becsülve nagyon hanyagok vagyunk. További jelölések segítik azt, hogy a pontos nagyságrendet kiemeljük.

Definíció

Legyen $t, f : \mathbb{N} \to \mathbb{R}$. Ekkor $t = \Omega(f)$ azt jelenti, hogy alkalmas $c > 0$ konstansra és n_0 küszöbértékre fennáll, hogy $n > n_0$ esetén

$$cf(n) \leq t(n).$$

- $f(n)$ mindig egy egy idő után pozitív függvény lesz.

Theta jelölés

Definíció

Legyen $t, f : \mathbb{N} \to \mathbb{R}$. Ekkor $t = \Theta(f)$ azt jelenti, hogy alkalmas $c, c' > 0$ konstansra és n_0 küszöbértékre fennáll, hogy $n > n_0$ esetén

$$cf(n) \leq t(n) \leq c'f(n)$$

Példa

Példa

Példa

$$18\binom{n}{6} + 127n^4 \log n + \log^{15}(n^{124}) \cdot n + 144 = \Theta(n^6).$$

Aszimptotikus jelölés

Aszimptotikus jelölés

Ha a nagyságrendet leíró függvény mellöl nem szeretnénk szönyeg alá söpörni a konstanst, akkor új jelölések szükségesek.

Aszimptotikus jelölés

Ha a nagyságrendet leíró függvény mellöl nem szeretnénk szönyeg alá söpörni a konstanst, akkor új jelölések szükségesek.

Definíció

$t(n) \sim f(n)$, azt jelenti, hogy $\lim_{n\to\infty} t(n)/f(n) = 1$.

Aszimptotikus jelölés

Ha a nagyságrendet leíró függvény mellöl nem szeretnénk szönyeg alá söpörni a konstanst, akkor új jelölések szükségesek.

Definíció

$t(n) \sim f(n)$, azt jelenti, hogy $\lim_{n\to\infty} t(n)/f(n) = 1$.

Másképpen $t(n) \sim f(n)$, azt jelenti, hogy $t(n) = f(n) + o(f(n))$.

Kis ordó jelölés

Kis ordó jelölés

Definíció

$g(n) = o(f(n))$ azt mondja a $g(n)$ függvényről, hogy $f(n)$-nel osztva 0-hoz tart, ha n tart a végtelenbe.

Kis ordó jelölés

Definíció

$g(n) = o(f(n))$ azt mondja a $g(n)$ függvényről, hogy $f(n)$-nel osztva 0-hoz tart, ha n tart a végtelenbe.

Elég nagy n esetén a $o(f(n))$-nel jelölt maradéktag, $f(n)$-hez képest elhanyagolható.

Példa

Példa

$$18\binom{n}{6} + 127n^4 \log n + \log^{15}(n^{124}) \cdot n + 144 \sim \frac{18}{6!} n^6,$$

még pontosabban

$$18\binom{n}{6} + 127n^4 \log n + \log^{15}(n^{124}) \cdot n + 144 = \frac{18}{6!} n^6 + \mathcal{O}(n^5).$$

Vége van!

Köszönöm a figyelmet!