Salami tactics

Peter Hajnal

Bolyai Institute, University of Szeged, Hungary

2023 Fall

The basic idea

The basic idea

- This design technique can be applied for search problem

The basic idea

- This design technique can be applied for search problem
- We are given a set S.

The basic idea

- This design technique can be applied for search problem
- We are given a set S. We want to find a specific element of it, or verify that S doesn't contain that element.

The basic idea

- This design technique can be applied for search problem
- We are given a set S. We want to find a specific element of it, or verify that S doesn't contain that element.
- Instead of locating the goal element, we "fast" exclude as "many" elements from the search space as possible (we must guarantee that the thrown away elements don't contain the goal element).

The basic idea

- This design technique can be applied for search problem
- We are given a set S. We want to find a specific element of it, or verify that S doesn't contain that element.
- Instead of locating the goal element, we "fast" exclude as "many" elements from the search space as possible (we must guarantee that the thrown away elements don't contain the goal element).
- The best way to understand this scheme is to see examples.

The scheme by picture

Binary search

Binary search

Given the ordered sequence of n real numbers: $x_1 < x_2 < \ldots < x_n$. Also given a number g. Decide whether g is among our x_i's. If yes, then provide the index i for that $x_i = g$.

Binary search

Binary search

Given the ordered sequence of n real numbers: $x_1 < x_2 < \ldots < x_n$. Also given a number g. Decide whether g is among our x_i's. If yes, then provide the index i for that $x_i = g$.

We assume that $n = 2^k - 1$.

Binary search

Binary search

Given the ordered sequence of n real numbers: $x_1 < x_2 < \ldots < x_n$. Also given a number g. Decide whether g is among our x_i's. If yes, then provide the index i for that $x_i = g$.

We assume that $n = 2^k - 1$. Specially in the ordered sequence there is a median/middle element: $x_{2^{k-1}}$.

The algorithm

Binary search algorithm

The algorithm

Binary search algorithm

Given the ordered sequence of n real numbers: $x_1 < x_2 < \ldots < x_n$, where $n = 2^k - 1$, and a number, g: the "goal".

The algorithm

Binary search algorithm

Given the ordered sequence of n real numbers: $x_1 < x_2 < \ldots < x_n$, where $n = 2^k - 1$, and a number, g: the "goal".

(0) If $n = k = 1$, after comparing our only number and g we can answer the problem. Otherwise

The algorithm

Binary search algorithm

Given the ordered sequence of n real numbers: $x_1 < x_2 < \ldots < x_n$, where $n = 2^k - 1$, and a number, g: the "goal".

(0) If $n = k = 1$, after comparing our only number and g we can answer the problem. Otherwise

(1) Compare g and $x_{2^{k-1}}$: g ? $x_{2^{k-1}}$.

The algorithm

Binary search algorithm

Given the ordered sequence of n real numbers: $x_1 < x_2 < \ldots < x_n$, where $n = 2^k - 1$, and a number, g: the "goal".

(0) If $n = k = 1$, after comparing our only number and g we can answer the problem. Otherwise

(1) Compare g and $x_{2^{k-1}}$: $g \; ? \; x_{2^{k-1}}$.

($2_=$) In the case $g = x_{2^{k-1}}$ we are very lucky. STOP.

The algorithm

Binary search algorithm

Given the ordered sequence of n real numbers: $x_1 < x_2 < \ldots < x_n$, where $n = 2^k - 1$, and a number, g: the "goal".

(0) If $n = k = 1$, after comparing our only number and g we can answer the problem. Otherwise

(1) Compare g and $x_{2^{k-1}}$: g ? $x_{2^{k-1}}$.

($2_=$) In the case $g = x_{2^{k-1}}$ we are very lucky. STOP.

($2_<$) In the case $g < x_{2^{k-1}}$ we can throw away $x_{2^{k-1}}, x_{2^{k-1}+1}, x_{2^{k-1}+2}, \ldots, x_n$ from the search.

The algorithm

Binary search algorithm

Given the ordered sequence of n real numbers: $x_1 < x_2 < \ldots < x_n$, where $n = 2^k - 1$, and a number, g: the "goal".

(0) If $n = k = 1$, after comparing our only number and g we can answer the problem. Otherwise

(1) Compare g and $x_{2^{k-1}}$: g ? $x_{2^{k-1}}$.

($2_=$) In the case $g = x_{2^{k-1}}$ we are very lucky. STOP.

($2_<$) In the case $g < x_{2^{k-1}}$ we can throw away $x_{2^{k-1}}, x_{2^{k-1}+1}, x_{2^{k-1}+2}, \ldots, x_n$ from the search.

($2_>$) In the case $g < x_{2^{k-1}}$ we can throw away $x_1, \ldots, x_{2^{k-1}-2}, x_{2^{k-1}-1}, x_{2^{k-1}}$ from the search.

The algorithm

Binary search algorithm

Given the ordered sequence of n real numbers: $x_1 < x_2 < \ldots < x_n$, where $n = 2^k - 1$, and a number, g: the "goal".

(0) If $n = k = 1$, after comparing our only number and g we can answer the problem. Otherwise

(1) Compare g and $x_{2^{k-1}}$: g ? $x_{2^{k-1}}$.

($2_=$) In the case $g = x_{2^{k-1}}$ we are very lucky. STOP.

($2_<$) In the case $g < x_{2^{k-1}}$ we can throw away $x_{2^{k-1}}, x_{2^{k-1}+1}, x_{2^{k-1}+2}, \ldots, x_n$ from the search.

($2_>$) In the case $g < x_{2^{k-1}}$ we can throw away $x_1, \ldots, x_{2^{k-1}-2}, x_{2^{k-1}-1}, x_{2^{k-1}}$ from the search.

// If we are here, then we reduced the search space to size of $2^{k-1} - 1$.

The algorithm

Binary search algorithm

Given the ordered sequence of n real numbers: $x_1 < x_2 < \ldots < x_n$, where $n = 2^k - 1$, and a number, g: the "goal".

(0) If $n = k = 1$, after comparing our only number and g we can answer the problem. Otherwise

(1) Compare g and $x_{2^{k-1}}$: g ? $x_{2^{k-1}}$.

($2_=$) In the case $g = x_{2^{k-1}}$ we are very lucky. STOP.

($2_<$) In the case $g < x_{2^{k-1}}$ we can throw away $x_{2^{k-1}}, x_{2^{k-1}+1}, x_{2^{k-1}+2}, \ldots, x_n$ from the search.

($2_>$) In the case $g < x_{2^{k-1}}$ we can throw away $x_1, \ldots, x_{2^{k-1}-2}, x_{2^{k-1}-1}, x_{2^{k-1}}$ from the search.

// If we are here, then we reduced the search space to size of $2^{k-1} - 1$.

(3) $k \leftarrow k - 1$. Return to (0).

Summary

Summary

The above algorithm uses k comparisons if g is not among our numbers.

Summary

The above algorithm uses k comparisons if g is not among our numbers. If g is one of the x_i-s it might stop earlier.

Summary

The above algorithm uses k comparisons if g is not among our numbers. If g is one of the x_i-s it might stop earlier.

Theorem

The binary search algorithm solves the problem using

$$\mathcal{O}(\log n)$$

comparison.

A popular form of binary search

A popular form of binary search

- We have s, a secret number from the set $\{1, 2, 3, \ldots, 99, 100\}$.

A popular form of binary search

- We have s, a secret number from the set $\{1, 2, 3, \ldots, 99, 100\}$.

- By asking Boolean questions about s (yes/no answer) figure out the value of s.

A popular form of binary search

- We have s, a secret number from the set $\{1, 2, 3, \ldots, 99, 100\}$.
- By asking Boolean questions about s (yes/no answer) figure out the value of s.
- How many questions are needed to achieve this goal?

Break

The basic problem

The basic problem

Definition: The median problem

Given n numbers: $x_1, x_2, \ldots, x_n$. Find the index m: x_m is the median of our numbers.

A generalization

A generalization

Definition: Generalized median problem

Given n numbers: $x_1, x_2, \ldots, x_n$, and a rank number $r \in \{1, 2, \ldots, n-1, n\}$. Classify the set of indices into three classes, $S \dot{\cup} \{m\} \dot{\cup} B = \{1, 2, \ldots, n-1, n\}$, the following way:

(1) for each $s \in S$ we have $x_s \leq x_m$,

(2) for each $b \in B$ we have $x_m \leq x_b$,

(3) $|S| = r - 1$.

A generalization

Definition: Generalized median problem

Given n numbers: $x_1, x_2, \ldots, x_n$, and a rank number $r \in \{1, 2, \ldots, n-1, n\}$. Classify the set of indices into three classes, $S \dot{\cup} \{m\} \dot{\cup} B = \{1, 2, \ldots, n-1, n\}$, the following way:

(1) for each $s \in S$ we have $x_s \leq x_m$,

(2) for each $b \in B$ we have $x_m \leq x_b$,

(3) $|S| = r - 1$.
// Hence $|B| = n - r$.

A generalization

Definition: Generalized median problem

Given n numbers: $x_1, x_2, \ldots, x_n$, and a rank number $r \in \{1, 2, \ldots, n-1, n\}$. Classify the set of indices into three classes, $S \dot{\cup} \{m\} \dot{\cup} B = \{1, 2, \ldots, n-1, n\}$, the following way:

(1) for each $s \in S$ we have $x_s \leq x_m$,

(2) for each $b \in B$ we have $x_m \leq x_b$,

(3) $|S| = r - 1$.
// Hence $|B| = n - r$.

Note that the median problem is the case when $r = \lfloor (n+1)/2 \rfloor$, furthermore we are not interested in the sets S and B.

A generalization

Definition: Generalized median problem

Given n numbers: $x_1, x_2, \ldots, x_n$, and a rank number $r \in \{1, 2, \ldots, n-1, n\}$. Classify the set of indices into three classes, $S \dot\cup \{m\} \dot\cup B = \{1, 2, \ldots, n-1, n\}$, the following way:

(1) for each $s \in S$ we have $x_s \leq x_m$,

(2) for each $b \in B$ we have $x_m \leq x_b$,

(3) $|S| = r - 1$.
// Hence $|B| = n - r$.

Note that the median problem is the case when $r = \lfloor (n+1)/2 \rfloor$, furthermore we are not interested in the sets S and B.

Again we have a simplifying assumption: n has the form $10k + 5 = (2k+1) \cdot 5$

The algorithm

The algorithm

Generalized median finding algorithm

The algorithm

Generalized median finding algorithm

Given n numbers, and a rank r. // Think about the n input numbers as they are arranged in a matrix of size $5 \times (2k + 1)$.

The algorithm

Generalized median finding algorithm

Given n numbers, and a rank r. // Think about the n input numbers as they are arranged in a matrix of size $5 \times (2k + 1)$.

(Column medians) For each columns (number quintets) determine the median and their S, B sets. (m_i is the median of the ith column, furthermore S_i, B_i are the corresponding sets.)

The algorithm

Generalized median finding algorithm

Given n numbers, and a rank r. // Think about the n input numbers as they are arranged in a matrix of size $5 \times (2k+1)$.

(Column medians) For each columns (number quintets) determine the median and their S, B sets. (m_i is the median of the ith column, furthermore S_i, B_i are the corresponding sets.)

(Median of medians) For the $n/5 = 2k+1$ medians find the median: μ, and the corresponding sets: $\widetilde{S}$, $\widetilde{B}$.

The algorithm

Generalized median finding algorithm

Given n numbers, and a rank r. // Think about the n input numbers as they are arranged in a matrix of size $5 \times (2k+1)$.

(Column medians) For each columns (number quintets) determine the median and their S, B sets. (m_i is the median of the ith column, furthermore S_i, B_i are the corresponding sets.)

(Median of medians) For the $n/5 = 2k+1$ medians find the median: μ, and the corresponding sets: $\widetilde{S}$, $\widetilde{B}$. // Note the following set contains only numbers that are not bigger than μ:

$$\widehat{S} = \bigcup \{\{m_i\} \cup S_i : m_i \in \widetilde{S}\} \cup S_i : m_i = \mu.$$

The algorithm

Generalized median finding algorithm

Given n numbers, and a rank r. // Think about the n input numbers as they are arranged in a matrix of size $5 \times (2k+1)$.

(Column medians) For each columns (number quintets) determine the median and their S, B sets. (m_i is the median of the ith column, furthermore S_i, B_i are the corresponding sets.)

(Median of medians) For the $n/5 = 2k+1$ medians find the median: μ, and the corresponding sets: $\widetilde{S}$, $\widetilde{B}$. // Note the following set contains only numbers that are not bigger than μ:

$$\widehat{S} = \bigcup \{\{m_i\} \cup S_i : m_i \in \widetilde{S}\} \cup S_i : m_i = \mu.$$

// The following set contains only numbers that are not smaller than μ:

$$\widehat{B} = \bigcup \{\{m_i\} \cup B_i : m_i \in \widetilde{B}\} \cup \{B_i : m_i = \mu\}.$$

The algorithm (cont'd)

The algorithm (cont'd)

Generalized median finding algorithm (cont'd)

The algorithm (cont'd)

Generalized median finding algorithm (cont'd)

(Finding the rank of μ) Compare μ to all other elements. Determine the corresponding S_μ, B_μ sets.

The algorithm (cont'd)

Generalized median finding algorithm (cont'd)

(Finding the rank of μ) Compare μ to all other elements. Determine the corresponding S_μ, B_μ sets.

$\rho = |S_\mu| + 1$.

The algorithm (cont'd)

Generalized median finding algorithm (cont'd)

(Finding the rank of μ) Compare μ to all other elements. Determine the corresponding S_μ, B_μ sets.

$\rho = |S_\mu| + 1$.

// We have $\widehat{S} \subset S_\mu$, and $\widehat{B} \subset B_\mu$.

The algorithm (cont'd)

Generalized median finding algorithm (cont'd)

(Finding the rank of μ) Compare μ to all other elements. Determine the corresponding S_μ, B_μ sets.

$\rho = |S_\mu| + 1$.

// We have $\widehat{S} \subset S_\mu$, and $\widehat{B} \subset B_\mu$.

(The cut of the salami) If $\rho = r$, then we are extremely lucky.

The algorithm (cont'd)

Generalized median finding algorithm (cont'd)

(Finding the rank of μ) Compare μ to all other elements. Determine the corresponding S_μ, B_μ sets.

$\rho = |S_\mu| + 1$.

// We have $\widehat{S} \subset S_\mu$, and $\widehat{B} \subset B_\mu$.

(The cut of the salami) If $\rho = r$, then we are extremely lucky.

If $\rho < r$, then throw away the numbers of $\widehat{S}$. $r \leftarrow r - |\widehat{S}|$.

The algorithm (cont'd)

Generalized median finding algorithm (cont'd)

(Finding the rank of μ) Compare μ to all other elements. Determine the corresponding S_μ, B_μ sets.

$\rho = |S_\mu| + 1$.

// We have $\widehat{S} \subset S_\mu$, and $\widehat{B} \subset B_\mu$.

(The cut of the salami) If $\rho = r$, then we are extremely lucky.

If $\rho < r$, then throw away the numbers of $\widehat{S}$. $r \leftarrow r - |\widehat{S}|$.

If $\rho > r$, then throw away the numbers of $\widehat{B}$.

The algorithm (cont'd)

Generalized median finding algorithm (cont'd)

(Finding the rank of μ) Compare μ to all other elements. Determine the corresponding S_μ, B_μ sets.

$\rho = |S_\mu| + 1$.

// We have $\widehat{S} \subset S_\mu$, and $\widehat{B} \subset B_\mu$.

(The cut of the salami) If $\rho = r$, then we are extremely lucky.

If $\rho < r$, then throw away the numbers of $\widehat{S}$. $r \leftarrow r - |\widehat{S}|$.

If $\rho > r$, then throw away the numbers of $\widehat{B}$.

(Iteration) If the cardinality of the leftover numbers is smaller or equal to five then solve the problem by brute force.

The algorithm (cont'd)

Generalized median finding algorithm (cont'd)

(Finding the rank of μ) Compare μ to all other elements. Determine the corresponding S_μ, B_μ sets.

$\rho = |S_\mu| + 1$.

// We have $\widehat{S} \subset S_\mu$, and $\widehat{B} \subset B_\mu$.

(The cut of the salami) If $\rho = r$, then we are extremely lucky.

If $\rho < r$, then throw away the numbers of $\widehat{S}$. $r \leftarrow r - |\widehat{S}|$.

If $\rho > r$, then throw away the numbers of $\widehat{B}$.

(Iteration) If the cardinality of the leftover numbers is smaller or equal to five then solve the problem by brute force.

Otherwise do the same for the leftover numbers, and r.

A claim

A claim

Claim

Let $n = (2k+1) \cdot 5$ as in the algorithm. Then

$$|\widehat{S}| = |\widehat{B}| = 3k + 2 = 3n/10 + 1/2.$$

A claim

Claim

Let $n = (2k+1) \cdot 5$ as in the algorithm. Then

$$|\widehat{S}| = |\widehat{B}| = 3k + 2 = 3n/10 + 1/2.$$

A claim

Claim

Let $n = (2k+1) \cdot 5$ as in the algorithm. Then

$$|\widehat{S}| = |\widehat{B}| = 3k + 2 = 3n/10 + 1/2.$$

Corollary

Let $n = (2k+1) \cdot 5$ as in the algorithm. Then in the first iteration we reduced the input size to

$$n - 3k + 2 = n - (3n/10 + 1/2) = 7n/10 - 1/2.$$

The analysis of the algorithm: Introduction

The analysis of the algorithm: Introduction

Let $M(n)$ be the maximal number of comparisons needed for our algorithm to solve the problem on n numbers.

The analysis of the algorithm: Introduction

Let $M(n)$ be the maximal number of comparisons needed for our algorithm to solve the problem on n numbers.

$M(x) = M(\lceil x \rceil)$.

The analysis of the algorithm: Introduction

Let $M(n)$ be the maximal number of comparisons needed for our algorithm to solve the problem on n numbers.

$M(x) = M(\lceil x \rceil)$.

Observation

$M(x)$ is a monotone increasing function.

The analysis of the algorithm

The analysis of the algorithm

Theorem

$$M(n) \leq M((n+9)/5) + M(7n/10) + O(n)$$

The analysis of the algorithm

Theorem

$$M(n) \leq M((n+9)/5) + M(7n/10) + O(n)$$

The third term counts the comparisons needed for (Column medians) and for (Finding the rank of μ).

The analysis of the algorithm

Theorem

$$M(n) \leq M((n+9)/5) + M(7n/10) + O(n)$$

The third term counts the comparisons needed for (Column medians) and for (Finding the rank of μ).

Theorem

$$M(n) \leq M(0.21 \cdot n) + M(0.7 \cdot n) + O(n),$$

assuming $n > 200$

The analysis of the algorithm

Theorem

$$M(n) \leq M((n+9)/5) + M(7n/10) + O(n)$$

The third term counts the comparisons needed for (Column medians) and for (Finding the rank of μ).

Theorem

$$M(n) \leq M(0.21 \cdot n) + M(0.7 \cdot n) + O(n),$$

assuming $n > 200$

Corollary

$$M(n) = O(n),$$

That is the end!

Thank you for your attention!