Binary Search and its Applications

Menghui Wang

1 Nov, 2011

Outline

Outline

Example: finding a value

Problem

You have an *ordered* sequence of numbers, $a_1, a_2, \ldots, a_n$.
Given x, decide whether x is in the sequence or not.

Example

$x = 12$.

3	7	12	44	53	76	98
low	mid	high lo mid	mid			high

Solution

We always focus on the solution interval, i.e., interval that the solution *must* fall in.

Each time, we pick a median a_m of the interval and compare it with x.

If $a_m = x$, we are lucky.

If $a_m < x$, we can conclude that the solution must be located at or right to a_{m+1}, since the numbers left to a_m are less than or equal to a_m, and thus smaller than our target.

The situation is similar when $a_m > x$.

Therefore, we can at least halve our interval each time.

Algorithm

```
low ← 1
high ← n
while low<=high do
    mid ← (low+high)/2
    if a[mid]= x then
        return  mid
    else if a[mid]< x then
        low ← mid+1
    else
        high ← mid-1
    end if
end while
complain
```

Outline

Finding a lower bound

Problem

You have an *ordered* sequence of numbers, $a_1, a_2, \ldots, a_n$.
Given x, find the *minimum* i such that $x \leq a_i$.
If such i doesn't exist, report.

Analysis

We focus on the solution interval as previous.

Each time, we pick a median a_m.

If $x <= a_m$, the solution must be located at or left to a_m.

Otherwise, the solution must be located at or right to a_{m+1}.

Algorithm for lower bound

```
low ← 1
high ← n
while low < high do
    mid ← (low+high)/2
    if x <=a[mid] then
        high ← mid
    else
        low ← mid+1
    end if
end while
if x <=a[low] then
    return low
else
    report no such i
end if
```

Finding a upper bound

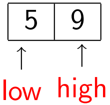
Problem

You have an *ordered* sequence of numbers, $a_1, a_2, \ldots, a_n$.
Given x, find the *maximum* i such that $a_i < x$.
If such i doesn't exist, report.

Algorithm for upper bound(WRONG!)

```
low ← 1
high ← n
while low < high do
    mid ← (low+high)/2
    if a[mid]< x then
        low ← mid
    else
        high ← mid-1
    end if
end while
if a[low]< x then
    return low
else
    report no such i
end if
```

Consider the following instance with $x = 7$.



Fixed algorithm for upper bound

```
low ← 1
high ← n
while low < high do
    mid ← (low+high+1)/2
    if a[mid]< x then
        low ← mid
    else
        high ← mid-1
    end if
end while
if a[low]< x then
    return low
else
    complain
end if
```

Outline

Any monotonic function will do!

We don't have to restrict binary search on ordered sequences.

Any *monotonic* function will do.

For example, a function that satisfies $f(x) \leq f(y)$ for all $x < y$.

In some cases, we need to construct such f and use binary search to to find the answer.

Outline

Union of intervals (SRM277)

Problem

Given a list of integers, find the n-th smallest number.
The numbers are given in intervals. For example, the intervals $[1, 3]$, $[2, 4]$ represent the list $\{1, 2, 3, 2, 3, 4\}$.
There are at most 50 intervals. For each interval $[a, b]$, $-2 \times 10^9 \leq a, b \leq 2 \times 10^9$.

Example

Given 2 intervals $[1, 4]$, $[3, 5]$ and $n = 4$.
From the intervals we know the numbers are $1, 2, 3, 3, 4, 4, 5$. The 4th smallest number is 3, so the answer is 3.

Solution

Let $f(x) = 1$ if "there are at least n numbers in the list that are less than or equal to x"; otherwise $f(x) = 0$.

For a certain x, $f(x)$ can be computed quickly.

$f(x)$ is monotonic.

The answer is the smallest x such that $f(x) = 1$.

Reduced to a lower bound problem.

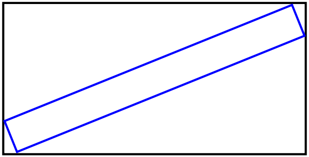
Outline

Packing rectangles (SRM270)

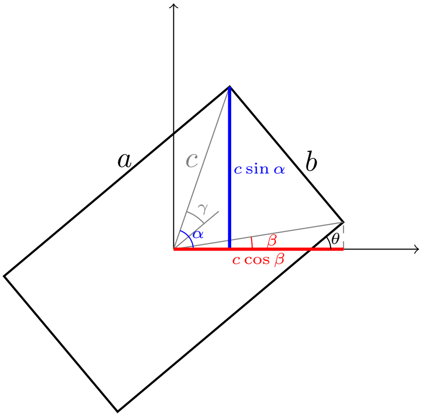
Problem

Given heights and lengths of two rectangles, decide whether the second rectangle fits in the first one.
Rotations are allowed.

Example



Solution



Suppose we rotate the second rectangle around its center by θ.

$\alpha = \theta + \gamma, \beta = \theta - \gamma$.

We can check if the rectangle fits by comparing $c \sin \alpha$ and $c \cos \beta$ with the first rectangle's height and length.

Inspect the monotonicity.