Binary Search

Ftan Camp 2018

February 15, 2018

Binary Search

```
// Input: [0,0,....,0,1,....,1,1]
// Output: Index of last 0, or -1 if there is none
int search(vector<int> const& a) {
  // Invariant: l==-1 or a[l]=0
  int l=-1, r=a.size();
  while (r - l > 1) {
    int m = l + (r - l)/2;
    if (a[m] == 0)
      l = m;
    else
      r = m;
  }
  return l;
}
```

Binary Search

Implicit representation of array with predicates.

$$P(x) \wedge (x < y) \implies P(y)$$

Example: Find index of value x in array $a_0, \ldots, a_{n-1}$

Binary Search

Implicit representation of array with predicates.

$$P(x) \wedge (x < y) \implies P(y)$$

Example: Find index of value x in array $a_0, \ldots, a_{n-1}$

$$P(i) = (a[i] > x)$$
$$P(-1) = 0$$

Sentinels

Make sure that the list starts with 0 and ends with 1.

Add $-\infty$ to the front and ∞ to the end to avoid any problems.

Example 1

N employees

K filing cabinets that contain $a_0 ..., a_{k-1}$ documents

Assign employee i to consecutive cabinets $l_i, ..., r_i$

All cabinets need to be covered by exactly one employee

Minimize $\max_i\{a_{l_i} + a_{l_i+1} + \cdots + a_{r_i-1}\}$

Example 3: Take Away

Reduction to a simpler problem using binary search:

What is the minimum cost?

$$\Longrightarrow$$

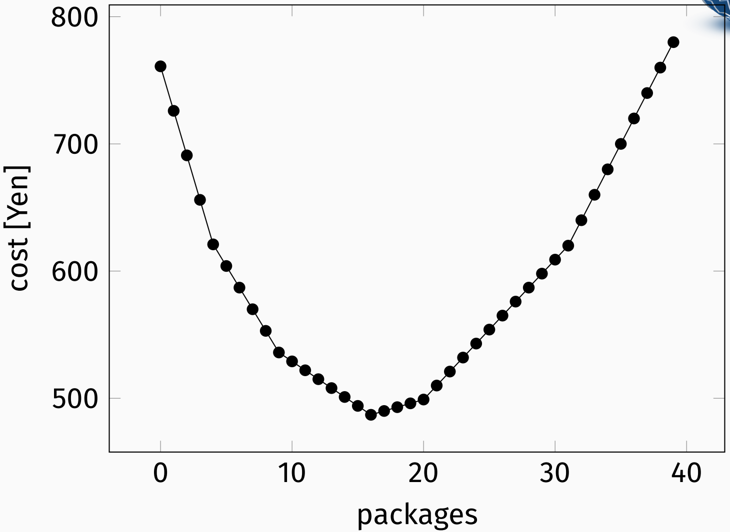
Can it be done with cost x?

Example 2

Simple skewness of a collection of numbers: mean − median.

Given a list of n integers (not necessarily distinct), find the non-empty subsequence with the maximum simple skewness.

Search on Convex Function



Search on Convex Function

- Binary search on derivative
- Ternary search (if you can't compute the derivative)

Example 3

Given a sequence of n integers $a_1, a_2, ..., a_n$.

Determine a real number x such that the weakness of the sequence $a_1 - x, a_2 - x, \ldots, a_n - x$ is as small as possible.

Weakness: maximum value of the poorness over all segments (contiguous subsequences) of a sequence.

Poorness of a segment: absolute value of the sum of the elements of segment.

Your answer will be considered correct if its relative or absolute error doesn't exceed 10^{-6}

Example 3: Take Away

```
// Input: [0,0,....,0,1,....,1,1]
// Output: Index of last 0
int search(double l, double r, double EPS) {
  // Invariant: a[l]=0
  while (r-l > EPS) {
    double m = (l + r)/2;
    if (pred(m))
      l = m;
    else
      r = m;
  }
  return l;
}
```

Running Time Analysis

How many operations?

Running Time Analysis

How many operations?

$$\mathcal{O}(\log((r - l)/e))$$

About Errors

x: Solution

$\tilde{x}$: Approximation of x

Absolute error (difference to solution?):

$$|x - \tilde{x}|$$

Relative error (percentage of error?):

$$\frac{|x - \tilde{x}|}{|x|} = \left|\frac{\tilde{x}}{x} - 1\right|$$

About Errors

```python
def IsApproximatelyEqual(x, y, epsilon=1e-6):
  # Check absolute precision.
  if -epsilon <= x - y <= epsilon:
    return True

  # Is x or y too close to zero?
  if (-epsilon <= x <= epsilon or
      -epsilon <= y <= epsilon):
    return False

  # Check relative precision.
  return (-epsilon <= (x - y) / x <= epsilon or
          -epsilon <= (x - y) / y <= epsilon)
```

Binary Search on Logarithm

Intuitive idea: Faster convergence for binary search on exponent

After some math:

$$m = \sqrt{lr} = e^{\frac{1}{2}(\log(l)+\log(r))}$$

Some pointer: `http://codeforces.com/blog/entry/49189`

Binary Search on Double Representation

Doubles are discrete. They can be casted to 64 bit integers.

Find the best double to approximate a problem: Binary search!

Binary Search on Double Representation

```
int search(double l, double r) {
  for (;;) {
    uint64_t l_bits = *(uint64_t*)(&l);
    uint64_t r_bits = *(uint64_t*)(&r);
    if (r_bits - l_bits <= 1)
      return l;
    uint64_t m_bits = l_bits+((r_bits-l_bits)>>1);
    double m = *(double*)(&m_bits);
    if (pred(m))
      l = m;
    else
      r = m;
  }
}
```

More on errors

Minimize absolute *and* relative error?

- Minimize absolute if $x < 1$.
- Minimize relative if $x > 1$.

Absolute error:

$$|x - \tilde{x}|$$

Relative error:

$$\frac{|x - \tilde{x}|}{|x|} = \left|\frac{\tilde{x}}{x} - 1\right|$$

How to find the best EPS

Needs to be larger than in task description because of rounding errors.

Can't be too large because the code will be too slow.

Hack: Fixed number of iterations!

No Upper Bound

Sometimes, there is no upper bound or you want your solution to be in $\mathcal{O}(log(ans))$.

```
int r = 1;
while (!pred(r))
  r *= 2;
return search(l, r);
```

Binary Search Tree

Can visualize binary search as a tree.

We look for the value that minimizes $(\text{Pred}(x), -x)$.

Branch and Bound: Define LB and UB => exactly the same as binary search.