HCPC 勉強会 (2016/06/30)

「二分探索をはじめからていねいに」

北海道大学情報エレクトロニクス学科
情報理工学コースB3 杉江祐哉

二分探索とは？

・やりたいこと

解の存在範囲を狭めていくことによって
最適解を求めたい！

・一回の操作で存在範囲は半分の範囲に

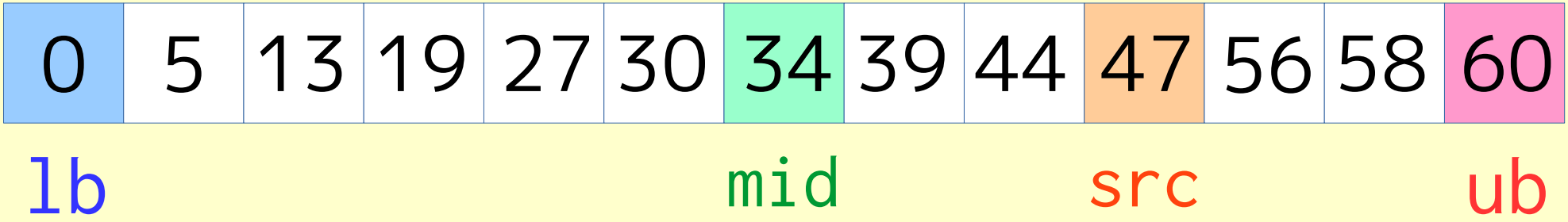
・計算量 O(logn)

二分探索のイメージ

0	5	13	19	27	30	34	39	44	47	56	58	60

- 昇順ソート済みの配列 a の要素について探索することを考える
- 今回は、47がどこにあるかを探してみよう

二分探索のイメージ




```
int n = 13;   //要素数
int src = 47; //探したい数字

int lb = 0;   //lower bound(下限)
int ub = n-1; //upper bound(上限)
int mid = (lb + ub) / 2;
```

二分探索のイメージ

midの要素は47以下である

→ 探索したい数字は配列の右半分にある！

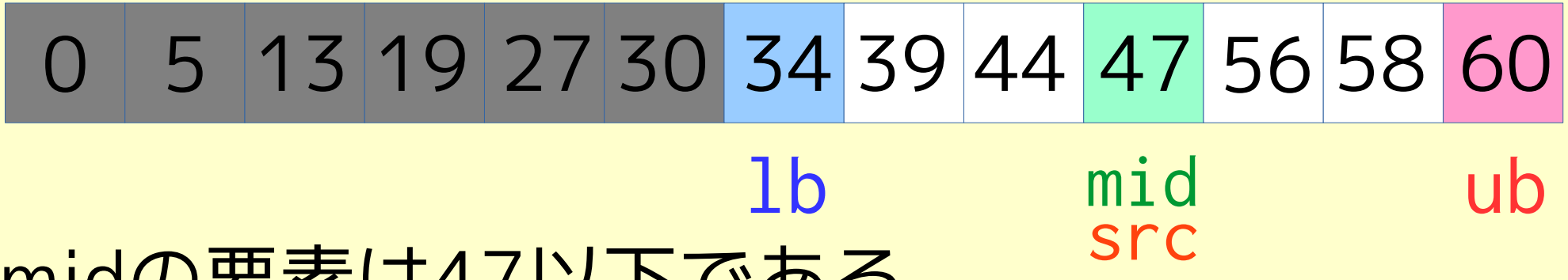
→ 探索範囲を右半分に絞る

※midの要素が47を超える数字の場合は、

探索範囲を左半分に絞る

これの繰り返しを、ub - lbが1になるまで行う

二分探索のイメージ



midの要素は47以下である

→ 探索したい数字は配列の右半分にある！

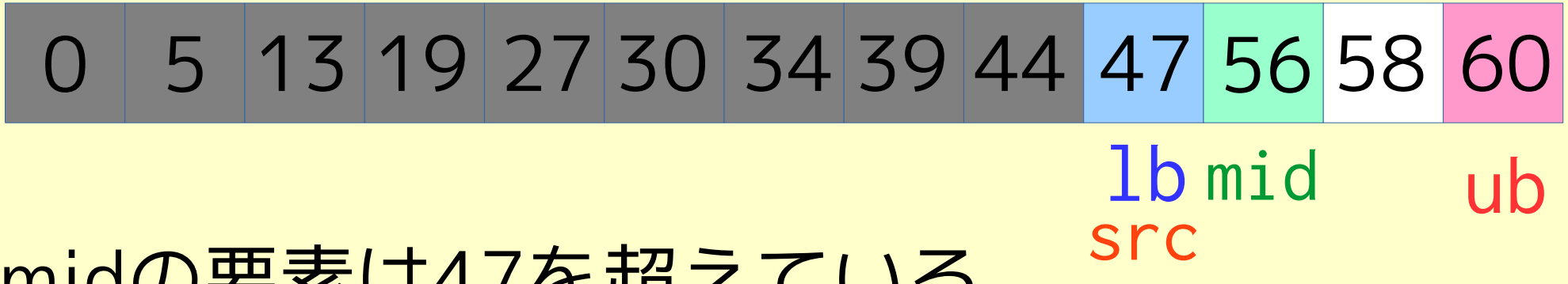
→ 探索範囲を右半分に絞る

※midの要素が47を超える数字の場合は、

探索範囲を左半分に絞る

これの繰り返しを、ub - lbが1になるまで行う

二分探索のイメージ



midの要素は47を超えている

→ 探索したい数字は配列の左半分にある！

→ 探索範囲を左半分に絞る

※midの要素が47以下である数字の場合は、

探索範囲を右半分に絞る

これの繰り返しを、ub - lbが1になるまで行う

二分探索のイメージ


```
while(ub-lb > 1) {
    if(a[mid] <= src) {
        lb = mid;
        mid = (lb + ub) / 2;
    }
    else if(a[mid] > src) {
        ub = mid;
        mid = (lb + ub) / 2;
    }
}
```

lbは、探索する要素

「以下」になる

最後の要素を指す(注意)

ubは、探索する要素を

「超える」

最初の要素を指す

二分探索 (C++)

- 先ほどの操作をしてくれるSTLが用意されている
- 要 #include <algorithm>
- イテレータが返ることに注意

```
int x;
int y;
x = *lower_bound(a.begin(), a.end(), src); //47
y = *upper_bound(a.begin(), a.end(), src); //56
```

- **lower_bound**は、探索する要素「以上」になる
  最初の要素のイテレータを取得する
- **upper_bound**は、探索する要素を「超える」
  最初の要素のイテレータを取得する

二分探索問題解説

二分探索は、今まで説明してきた
「値の検索」以外の使い方もできる

…今回は4題を解説！

- Cable master
- Aggressive cows
- Monthly Expense
- 平均最大化

Cable master

長さがそれぞれL_iであるようなN本の紐があります。これらの紐を切って、同じ長さの紐をK本作るときの最長の長さを求めなさい。答えは小数点以下2桁までを切り捨てで出力しなさい。

制約:

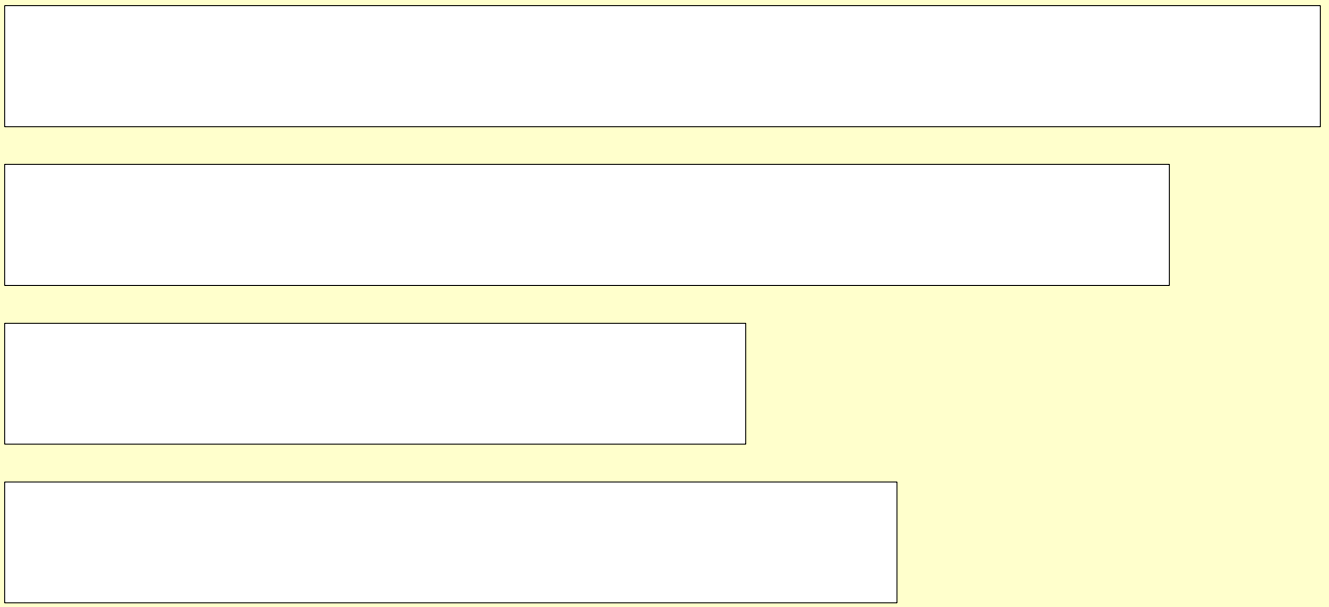
- $1 \leqq N \leqq 10,000$
- $1 \leqq K \leqq 10,000$
- $1 \leqq L_i \leqq 100,000$

Cable master

たとえば…

N = 4, K = 11,

L = {8.02, 7.43, 4.57, 5.39}



Cable master

- ここで復習…
  - 二分探索は、解の存在範囲を狭めていくことによって最適解を求める方法！

- 今回の問題は…
  - 同じ長さの紐は、切り出す長さが短いほどたくさん作れ、長いほど作れなくなる
  - 切り出せる長さの存在範囲を狭めていこう！

Cable master

条件 C(x) = 「長さxの紐をK本切り出すことができるかどうか (bool値)」 を用意

```
bool C(double x) {
    int ret = 0;
    for(int i=0; i<N; i++) {
        ret += floor(L[i] / x);
    }
    return ret >= K;
}
```

※floor関数は、切り捨て操作を行う関数

Cable master

今回は、ub – lbの値を終了条件にするのではなく、

二分探索をした回数を終了条件とする

```
void solve() {
    double lb = 0, ub = INF;
    for(int i=0; i<100; i++) {
        double mid = (lb + ub) / 2;
        if(C(mid)) lb = mid;
        else ub = mid;
    }
    printf("%.2f\n", floor(ub * 100) / 100);
}
```

※二分探索を100回行うと、解の最終的な存在範囲は、
(最初の範囲) × (½) ^ 100 になるため、十分範囲が狭くなる！

Aggressive cows

農夫ジョンは、N個の牛舎を持つ小屋を作りました。各牛舎は直線上に並んでおり、i番目の牛舎は位置x_iにあります。しかし、彼のM頭の牛達は小屋の配置が気に入らず、とても攻撃的になってしまいました。ジョンは、牛達が互いに傷つけあうのを防ぐため、他の牛とできるだけ離れるように牛舎に入れることにしました。最も近い二頭の牛の間の間隔を最大化しなさい。

制約:

- $2 \leqq N \leqq 100,000$
- $2 \leqq M \leqq N$
- $1 \leqq x_i \leqq 10^9$

Aggressive cows

たとえば…

N = 5, M = 3,

x = {1, 2, 8, 4, 9}

1　　2　　4　　8　　9

最善で、(最も近い二頭の牛の間の間隔) = 3より、

答えは3

「最小値の最大化」問題！

Aggressive cows

条件 C(d) = 「最も近い二頭の牛の間隔をd以上にすることができる」を用意 (ソート前提) 。ちゃんと書けるようになろう

```
bool C(int d) {
    // 牛を入れることが確定している牛舎の中で、最も右の牛舎: last
    int last = 0;

    for(int i=1; i<M; i++) {
        // last 以降の位置にある牛舎: crt
        int crt = last + 1;
        // last と crt の間隔がd以上になるようなcrtを選ぶ
        while(crt < N && x[crt] - x[last] < d) crt++;

        // 間隔がd以上になるようなcrtが選べなかった
        if(crt == N) return false;
        // 選べたら、lastを更新する
        last = crt;
    }

    // M頭無事に牛舎に入れたらtrue
    return true;
}
```

Aggressive cows

条件さえちゃんと書ければ、後は普通の二分探索

```
void solve() {
    sort(x, x + N);

    int lb = 0, ub = INF;
    while(ub - lb > 1) {
        int mid = (lb + ub) / 2;
        if(C(mid)) lb = mid;
        else ub = mid;
    }
    cout << lb << endl;
}
```

Monthly Expense

農夫ジョンは経理の達人です。彼は農場の運営でお金がなくなってしまうかもしれないことに気がつきました。彼はすでに、1日に必要な金額v_iを、今後N日にわたって正確に計算・記録しています。

ジョンは”fajomonths”というM期の会計期間に分けて予算を作ろうとしています。会計期間は1日以上の連続する期間からなり、同じ日が複数の会計期間に含まれることはありません。この時、fajomonth中の出費の合計の最大値を最小化しなさい。

制約:

- $1 \leqq N \leqq 100,000$
- $1 \leqq M \leqq N$
- $1 \leqq v_i \leqq 10,000$

Monthly Expense

たとえば…

N = 7, M = 5

100	400	300	100	500	101	400

↓

100	400	300	100	500	101	400

区間の項の和の最大値は500なので、答えは 500

「最大値の最小化」問題！

Monthly Expense

条件 C(x) = 「区間にある項の和がx以下になる区間をM個作ることができる」を用意。

```
bool C(int x) {
    int cnt = 1, sum = 0;
    for(int i=0; i<n; i++) {
        if(v[i] > x) return false;
        // 足してもx以下の場合は、足す
        else if(sum + v[i] <= x) sum += v[i];
        // 足したらxを超える場合は、区切る
        else {
            cnt++;
            sum = v[i];
        }
    }
    // 区間の数がmを超えていなければ実現可能
    return cnt <= m;
}
```

Monthly Expense

これも条件さえ書けばいつものやつに

```
void solve() {
    int lb = 0, ub = INF;
    while(ub - lb > 1) {
        int mid = (lb + ub) / 2;
        if(C(mid)) ub = mid;
        else lb = mid;
    }
    cout << ub << endl;
}
```

平均最大化

重さと価値がそれぞれw_i,v_iであるようなn個の品物があります。この中からちょうどk個選んだ時の単位重さあたりの価値の最大値を求めなさい。

制約:

- $1 \leqq k \leqq n \leqq 10^4$
- $1 \leqq w_i, v_i \leqq 10^6$

平均最大化

たとえば…

n = 3, k = 2,

(w, v) = { (2 , 2) , (5 , 3) , (2 , 1) }

答えは (2 , 2) , (2 , 1) の2つを選んだ時の 0.75

※単位重さあたりの価値が大きい物から貪欲にとるのは間違い！

平均最大化

ここでも、条件 C(x)を考える

↓

「単位重さあたりの価値がx以上となるように取ることができる」

この判定はどうやって実現できるか？

平均最大化

ある品物の集合Sに対して、単位重さあたりの価値は

$$\frac{\sum_{i \in S} v_i}{\sum_{i \in S} w_i}$$

で表される。単位重さあたりの価値がx以上になれば良いので、

$$\frac{\sum_{i \in S} v_i}{\sum_{i \in S} w_i} \geqq x$$

が成立すればよい。

平均最大化

さっきの式:

$$\frac{\sum_{i \in S} v_i}{\sum_{i \in S} w_i} \geqq x$$

これは、以下のように変形できる。

$$\sum_{i \in S} (v_i - x \times w_i) \geqq 0$$

これならば($v_i - x \times w_i$)の値でソートして貪欲に選べるので、C(x) = 「($v_i - x \times w_i$) の大きい方からk個の和が0以上」として解ける。

平均最大化

条件 C(x) の実装

```
bool C(double x) {
    // 各品物についてv[i] - x * w[i]を計算
    for(int i=0; i<n; i++) {
        y[i] = v[i] - x * w[i];
    }
    // yを降順ソート
    sort(y, y+n, greater<double>());

    // yの大きい順にk個とって和を求める
    double sum = 0;
    for(int i=0; i<k; i++) {
        sum += y[i];
    }

    return sum >= 0;
}
```

平均最大化

solve部はCable masterとほぼ同じ

```
void solve() {
    double lb = 0, ub = INF;
    // Cable masterと同様に、二分探索の回数が終了条件
    for(int i=0; i<100; i++) {
        double mid = (lb + ub) / 2;
        if(C(mid)) lb = mid;
        else ub = mid;
    }
    printf("%.2f\n",ub);
}
```

まとめ

- 二分探索でやりたいこと
  - 解の存在範囲を半分に狭めていくことによって最適解を求めたい！
- どんな問題で二分探索が有効か？
  - 値の探索 (ソート済み配列、典型的な問題)
  - 最小値の最大化 (Aggressive cows)
  - 最大値の最小化 (Monthly Expense)
  - ある条件下での最大化、最小化 (Cable master, 平均最大化)
- 終了条件は？
  - while(ub – lb > 1) パターン (配列操作、答えが整数)
  - 二分探索の回数で判定するパターン (100回やれば十分)