

# アルゴリズム論 Theory of Algorithms

## 第4回講義 分割統治法と漸化式

1/44

# アルゴリズム論 Theory of Algorithms

## Lecture #4 Divide and Conquer and Recurrence Equation

2/44

### 分割統治法

問題をいくつかの部分問題に分解して、それぞれの部分問題を再帰的に解いた後、部分問題の解を利用して元の問題を解く。

- 分割**: 問題をいくつかの部分問題に分割する(2分割など)。
- 統治**: 部分問題を再帰的に解く。ただし、部分問題のサイズが小さいときは直接的に解く。
- 統合**: 部分問題の解を統合して全体の解を構成する。

プログラムは次のような形式:

```
function DQ(x){
  if 問題 x が十分に小さい then 特別の方法を適用して答を返す;
  問題 x をいくつかの部分問題  $x_1, x_2, \dots, x_k$  に分割;
  for i=1 to k
     $y_i = DQ(x_i)$ ; //部分問題  $x_i$  の解  $y_i$  を再帰的に求める;
  部分問題の解  $y_1, y_2, \dots, y_k$  を組み合わせて元の問題 x の
  解 y を得て, y を返す;
}
```

3/44

### Divide and Conquer

Decompose the problem into several subproblems, solve them recursively, and then find a solution to the original problem by combining the solutions to the subproblems.

- Divide**: Decompose the problem into several subproblems.
- Conquer**: Solve the subproblems recursively. If they are small enough, we solve them directly.
- Merge**: Combine those solutions to have a solution to the problem.

Program is of the following form:

```
function DQ(x){
  if problem x is small enough then apply an adhoc procedure;
  decompose x into several subproblems  $x_1, x_2, \dots, x_k$  ;
  for i=1 to k
     $y_i = DQ(x_i)$ ; //solve  $x_i$  recursively;
  combine the solutions  $y_1, y_2, \dots, y_k$  to obtain a solution y to the
  original problem x and return y;
}
```

4/44

### 問題P7: 配列に蓄えられたデータの最大値を求めよ。

最も単純なアルゴリズムは、順に配列要素を調べて、以前に求まっている最大値より大きい要素を見つければ最大値を更新するというもの。--->アルゴリズムP7-A0.

	0	1	2	3	4	5	6	7	8
a	17	32	19	22	28	16	18	20	39

```
アルゴリズムP7-A0:
max=a[0];
for(i=1; i<n; i++)
  if(a[i] > max) max = a[i];
cout << max;
```

5/44

### Problem P7: Find a largest value in an array.

In the most simple algorithm we check all the elements and if we find a larger one than the largest one so far then we update the largest value.--->algorithm P7-A0.

	0	1	2	3	4	5	6	7	8
a	17	32	19	22	28	16	18	20	39

```
Algorithm P7-A0:
max=a[0];
for(i=1; i<n; i++)
  if(a[i] > max) max = a[i];
cout << max;
```

6/44

**分割統治法に基づく最大値発見**

**分割:** 配列を左半分と右半分に分割。  
**統治:** 配列のサイズが1になれば、その配列要素を出力。  
**統合:** 左右の部分の最大値のうち大きい方を出力。

**アルゴリズムP7-A1:**

```
int FindMax(int left, int right){
    if(right==left) return a[left];
    int mid = (left+right)/2;
    int x1 = FindMax(left, mid);
    int x2 = FindMax(mid+1, right);
    if(x1>x2) return x1; else return x2;
}
main(){
    .....
    cout << FindMax(0, n-1);
}
```

サイズnの配列で最大値を求めるのに要する時間をT(n)とすると、  
 $T(1) = 1$ .  
 $T(n) \leq 2T(n/2)+3$ .  
 この漸化式を解くと  
 $T(n) = O(n)$ .

練習問題: 上の漸化式を解け。

7/44

**Finding Maximum based on Divide and Conquer**

**Divide:** Decompose array into two halves  
**Conquer:** If the array size is 1, output the element.  
**Merge:** Output the larger one of the two maximums.

**Algorithm P7-A1:**

```
int FindMax(int left, int right){
    if(right==left) return a[left];
    int mid = (left+right)/2;
    int x1 = FindMax(left, mid);
    int x2 = FindMax(mid+1, right);
    if(x1>x2) return x1; else return x2;
}
main(){
    .....
    cout << FindMax(0, n-1);
}
```

Let T(n) be time to find a maximum among n data, then we have  
 $T(1) = 1$ .  
 $T(n) \leq 2T(n/2)+3$ .  
 Solving it, we have  
 $T(n) = O(n)$ .

Exercise: Solve the above recurrence equation.

8/44

**分割統治法に基づく最大値発見**

配列を1個と残り全部に分けるとどうか?

**アルゴリズムP7-A2:**

```
int FindMax(int left, int right){
    if(right==left) return a[left];
    int x1 = FindMax(left, left);
    int x2 = FindMax(left+1, right);
    if(x1>x2) return x1; else return x2;
}
main(){
    .....
    cout << FindMax(0, n-1);
}
```

サイズnの配列で最大値を求めるのに要する時間をT(n)とすると、  
 $T(1) = 1$ .  
 $T(n) \leq T(1)+T(n-1)+2$ .  
 この漸化式を解くと  
 $T(n) = O(n)$ .

練習問題: 上の漸化式を解け。

つまり、最大値発見のような簡単な問題であれば、どちらも空でないように分割すれば、どのようにしても効率は殆んど同じ。

9/44

**Finding maximum based on divide and conquer**

What happens if we decompose an array into one and remaining?

**Algorithm P7-A2:**

```
int FindMax(int left, int right){
    if(right==left) return a[left];
    int x1 = FindMax(left, left);
    int x2 = FindMax(left+1, right);
    if(x1>x2) return x1; else return x2;
}
main(){
    .....
    cout << FindMax(0, n-1);
}
```

Let T(n) be time to find a maximum among n data, we have  
 $T(1) = 1$ .  
 $T(n) \leq T(1)+T(n-1)+2$ .  
 Solving it, we have  
 $T(n) = O(n)$ .

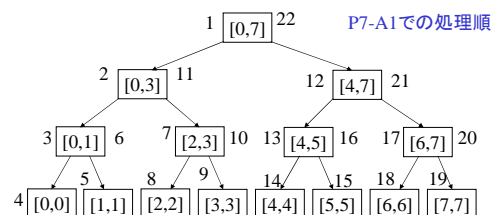
Exercise: Solve the above recurrence equation.

That is, for a simple problem of finding a maximum, the efficiency is almost the same if we decompose an array two non-empty parts.

10/44

**最大値を求めるアルゴリズム**

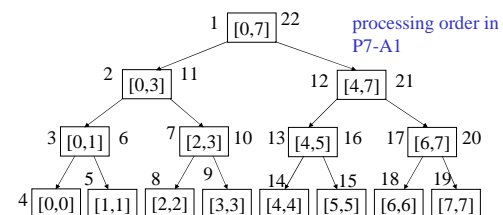
P7-A1: 配列を毎回ほぼ等しいサイズに分割。  
 P7-A2: 配列を1個と残り全部に分割。  
 どの方法も計算時間はO(n)で同じ。  
 何か違いはあるか?



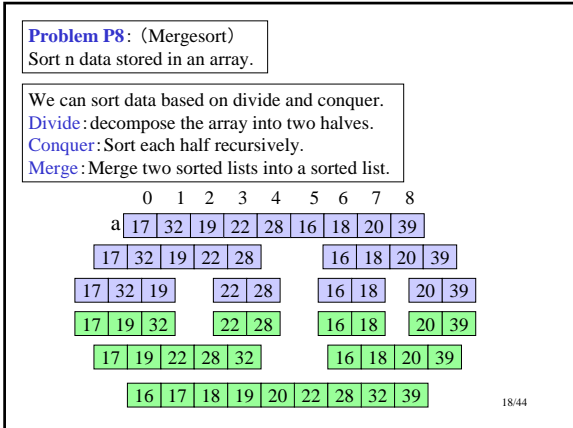
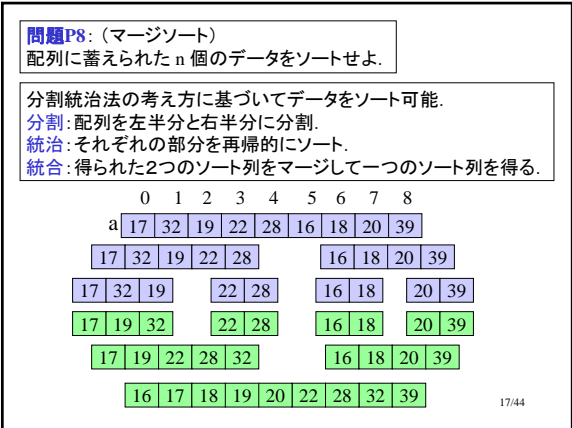
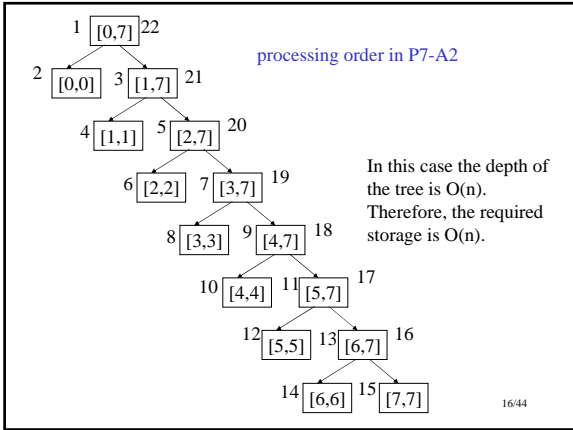
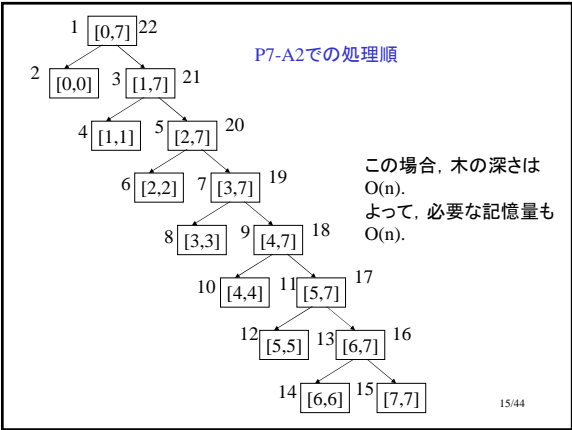
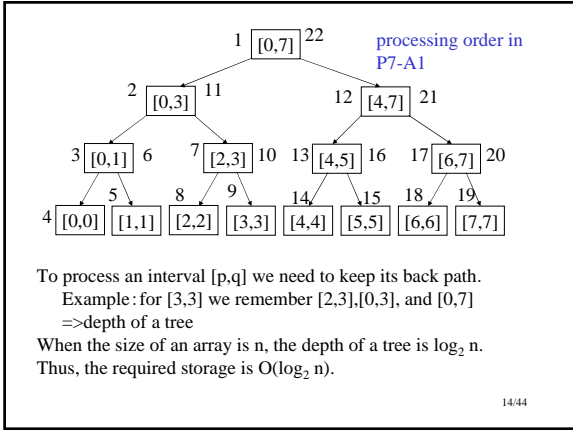
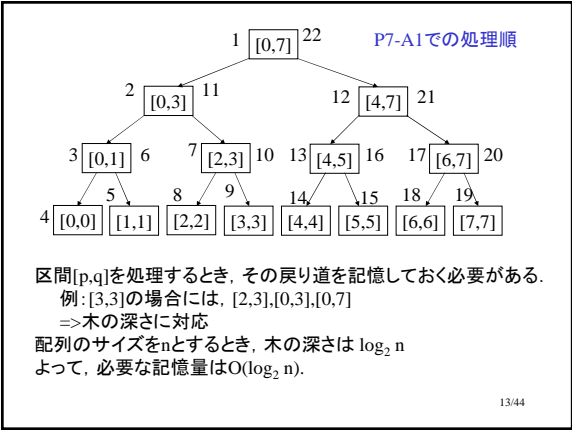
11/44

**Algorithm for finding a maximum**

P7-A1: decompose an array into two parts of the same length.  
 P7-A2: decompose an array into one and the remaining.  
 Both algorithms run in O(n) time.  
 Any difference between them?



12/44



**計算時間の解析**

n 個のデータをマージソートでソートするのに要する時間 (比較回数)をT(n)と書くことにする。  
 半分のサイズの問題を2回解いて、最後に2つのソート列をマージするが、マージは線形時間でできるから cn と置くと、  
 $T(n) \leq 2T(n/2) + cn$ ,  
 を得る。これを解けばよい。  
 $T(n) \leq 2T(n/2) + cn$   
 $\leq 2(2T(n/2^2)+c(n/2))+cn = 2^2T(n/2^2)+2cn$   
 $\leq 2^2(2T(n/2^3)+c(n/2^2))+2cn = 2^3T(n/2^3)+3cn$   
 $\leq \dots \leq 2^kT(n/2^k)+kcn$   
 ここで、 $2^k=n$ ,  $T(1)=$ 定数d, とすると、 $k=\log n$ だから、  
 $T(n) \leq dn + cn \log n = O(n \log n)$   
 を得る。

19/44

**Analysis of Computation Time**

Let T(n) be time (# of comparisons) for sorting n data by Mergesort. We solve the half-sized problems twice and then sort two sorted lists. Since merge operation is done in linear time, let it be cn. Then, we have  
 $T(n) \leq 2T(n/2) + cn$ ,  
 Solving it,  
 $T(n) \leq 2T(n/2) + cn$   
 $\leq 2(2T(n/2^2)+c(n/2))+cn = 2^2T(n/2^2)+2cn$   
 $\leq 2^2(2T(n/2^3)+c(n/2^2))+2cn = 2^3T(n/2^3)+3cn$   
 $\leq \dots \leq 2^kT(n/2^k)+kcn$ .  
 If we assume  $2^k=n$ ,  $T(1)=$ a constant d, then we have  $k=\log n$  and  
 $T(n) \leq dn + cn \log n = O(n \log n)$ .

20/44

**問題P9: (中央値選択)**

配列に蓄えられたn個のデータの中央値を求めよ。

0 1 2 3 4 5 6 7 8  
 a | 17 | 32 | 19 | 22 | 28 | 16 | 18 | 20 | 39 |

昇順に並べると、16,17,18,19,20,22,28,32,39  
 なので、中央値は20。  
 nが偶数なら、中央値は2つある。

一般には、n 個のデータの中の k 番目に大きいものを求める問題。

**アルゴリズムP9-A0:**

n 個のデータをO(n log n)時間でソート。  
 k 番目のデータを出力。

このアルゴリズムで正しくk 番目の要素が求まる。  
 しかし、O(n log n)の時間が必要だろうか？

21/44

**Problem P9: (Median finding)**

Find a median of n data stored in an array.

0 1 2 3 4 5 6 7 8  
 a | 17 | 32 | 19 | 22 | 28 | 16 | 18 | 20 | 39 |

increasing order: 16,17,18,19,20,22,28,32,39  
 thus, the median is 20.

Note that is n is even then there are two medians.

General problem is to find the k-th largest element among n data.

**Algorithm P9-A0:**

Sort n data in O(n log n) time.  
 Output the k-th element.

This algorithm always finds the k-th largest element.  
 But, does it really need O(n log n) time?

22/44

**分割統治法に基づく方法**

**アルゴリズムP9-A1:**

n 個のデータを配列a[]に蓄える。  
 一つの配列要素 x を適当に選び、配列を順に調べて、  
 x 以下の要素の集合Sと、x以上の要素の集合Lに分ける。  
 if  $k \leq |L|$  then  
   集合Lの中でk番目に大きい要素yを再帰的に求める。  
 else  
   集合Sの中でk-|L|番目に大きい要素yを再帰的に求める。  
 yを出力する。

0 1 2 3 4 5 6 7 8  
 a | 17 | 32 | 19 | 22 | 28 | 16 | 18 | 20 | 39 |  
 | 17 | 20 | 19 | 22 | 18 | 16 |    | 28 | 32 | 39 |  
   S  $\leq$  28                      L  $\geq$  28

23/44

**Algorithm based on Divide and Conquer**

**Algorithm P9-A1:**

Store n data in an array a[].  
 Choose an arbitrary element x and decompose n data into  
 a set S smaller or equal to x and a set L larger or equal to x.  
 if  $k \leq |L|$  then  
   recursively find the k-th largest element in the set L.  
 else  
   recursively find the (k-|L|)-th largest element in the set S.  
 Output y.

0 1 2 3 4 5 6 7 8  
 a | 17 | 32 | 19 | 22 | 28 | 16 | 18 | 20 | 39 |  
 | 17 | 20 | 19 | 22 | 18 | 16 |    | 28 | 32 | 39 |  
   S  $\leq$  28                      L  $\geq$  28

24/44

プログラム例

```
int Find_k_largest(int low, int high, int k)
{
    int s=low, t=high, x=a[(s+t)/2];
    while(s < t){
        while(a[s]>x) s++;
        while(a[t]<x) t--;
        if(s<t) swap(&a[s++], &a[t--]);
    }
    if(k <= t+1) find_k_largest(low, t, k);
    else if(k >= s) find_k_largest(s,high, k-s);
    else return x;
}
main()
{
    .....
    cout << find_k_largest(0,n-1,k);
    .....
}
```

25/44

An example of a program

```
int Find_k_largest(int low, int high, int k)
{
    int s=low, t=high, x=a[(s+t)/2];
    while(s < t){
        while(a[s]>x) s++;
        while(a[t]<x) t--;
        if(s<t) swap(&a[s++], &a[t--]);
    }
    if(k <= t+1) find_k_largest(low, t, k);
    else if(k >= s) find_k_largest(s,high, k-s);
    else return x;
}
main()
{
    .....
    cout << find_k_largest(0,n-1,k);
    .....
}
```

26/44

計算時間の解析

最悪の場合、長さnの区間を長さ1とn-1の2つの区間に分割することになる。長さnの区間を処理するのに必要な時間をT(n)とすると、

$$T(n) \leq T(1)+T(n-1)+cn$$

となる。これを解くと、

$$T(n) = O(n^2).$$

平均比較回数をC(n,k)とすると、

$$C(n,k)=n+1+(1/n)(\sum_{t=0 \sim k-2} C(n-t-1,k-t-1)+\sum_{t=k+1 \sim n-1} C(t+1,k))$$

この漸化式を解くと、

$$C(n,k) = O(n)$$

となり、平均的には線形時間で終わることが分かる。

練習問題: アルゴリズムから上記の漸化式を導け。

練習問題: 上記の漸化式を解け。

27/44

Analysis of Computation Time

Worst case: an interval of length n is decomposed into intervals of lengths 1 and n-1. If we denote by T(n) time for processing an interval of length n, we have

$$T(n) \leq T(1)+T(n-1)+cn.$$

Solving it, we have

$$T(n) = O(n^2).$$

If we denote the average number of comparisons by C(n,k),

$$C(n,k)=n+1+(1/n)(\sum_{t=0 \sim k-2} C(n-t-1,k-t-1)+\sum_{t=k+1 \sim n-1} C(t+1,k)).$$

Solving this recurrence equation, we have

$$C(n,k) = O(n),$$

which implies that the average running time of the algorithm is O(n).

Exercise: Obtain the recurrence equation from the algorithm.

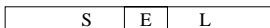
Exercise: Solve the above recurrence equation.

28/44

最悪の場合にも線形時間のアルゴリズム

アルゴリズムP9-A2:

- (1) n個のデータを15個ずつのグループに分割し、各グループごとに15個のデータの中央値を求める。
- (2) このようにして得られたn/15個の中央値に、この方法を再帰的に適用し、中央値の中央値Mを求める。
- (3) n個のデータをMに関して分割:  
S = Mより小さいデータの集合,  
L = Mより大きいデータの集合,  
E = Mに等しいデータの集合.
- (4)  $k \leq |L|$ のとき、Lの中でk番目に大きな要素を再帰的に求める。
- (5)  $k > |L|+|E|$ のとき、Sの中でk-|L|-|E|番目に大きな要素を再帰的に求める。
- (6) 上記以外の場合、Mが求める答である。

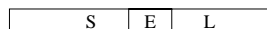


29/44

Linear-time Algorithm in the worst case

Algorithm P9-A2:

- (1) decompose n data into groups each containing at most 15 data and find the median in each group.
- (2) Find the median M of these n/15 medians obtained recursively.
- (3) Decompose the n data with respect to M:  
S = a set of data < M,  
L = a set of data > M,  
E = a set of data = M.
- (4) If  $k \leq |L|$ , find the k-th largest element in L recursively.
- (5) If  $k > |L|+|E|$ , find the (k-|L|-|E|)-th largest element in S recursively.
- (6) Otherwise, return M as a solution.



30/44

**例題**

: 24個のデータをサイズ5のグループに分割し、中央値を求める  
 $S = \{12, 56, 43, 22, 31, 25, 57, 75, 45, 33, 39, 85, 37, 44, 19, 28, 18, 23, 92, 73, 77, 28, 64, 35\}$   
 $S = \{12, 56, 43, 22, 31, 25, 57, 75, 45, 33, 39, 85, 37, 44, 19, 28, 18, 23, 92, 73, 77, 28, 64, 35\}$   
 中央値 = {31, 45, 39, 28, 35} 中央値の中央値  $M = 35$

35より大きい = {56, 43, 57, 75, 45, 39, 85, 37, 44, 92, 73, 77, 64} 13 要素 > 24/2  
 全体の中央値はこの集合にある。  
 この集合で12番目に大きい要素を再帰的に求める  
 $L = \{56, 43, 57, 75, 45, 39, 85, 37, 44, 92, 73, 77, 64\}$   
 中央値 = {56, 44, 73}, 中央値の中央値  $M = 56$   
 56より大きい = {57, 75, 85, 92, 73, 77, 64} 7 要素 > 13/2  
 12番目に大きい要素はこの集合にない  
 残りの集合の中で (12-7)番目に大きい要素を求める  
 $S = \{56, 43, 45, 39, 37, 44\}$  5番目に大きい要素 = 39  
 全体の中央値は 39

実際  
 > 39: 56, 43, 57, 75, 45, 85, 44, 92, 73, 77, 64,  
 39  
 < 39: 12, 22, 31, 25, 33, 37, 19, 28, 18, 23, 28, 35

31/44

**Example:** Decompose 24 data into groups of size 5 to find the median.

$S = \{12, 56, 43, 22, 31, 25, 57, 75, 45, 33, 39, 85, 37, 44, 19, 28, 18, 23, 92, 73, 77, 28, 64, 35\}$   
 $S = \{12, 56, 43, 22, 31, 25, 57, 75, 45, 33, 39, 85, 37, 44, 19, 28, 18, 23, 92, 73, 77, 28, 64, 35\}$   
 group medians = {31, 45, 39, 28, 35} the median of the medians  $M = 35$

data > 35 = {56, 43, 57, 75, 45, 39, 85, 37, 44, 92, 73, 77, 64} 13 elements > 24/2  
 The overall median must be in this set  
 Find the 12th largest element in this set recursively  
 $S = \{56, 43, 57, 75, 45, 39, 85, 37, 44, 92, 73, 77, 64\}$   
 group medians = {56, 44, 73}, the median of the medians  $M = 56$   
 data > 56 = {57, 75, 85, 92, 73, 77, 64} 7 elements > 13/2  
 The 12-th largest element cannot be in this set  
 Find the (12-7)-th largest element in the remaining set.  
 $S = \{56, 43, 45, 39, 37, 44\}$  5-th largest element = 39  
 The overall median is 39

In fact,  
 > 39: 56, 43, 57, 75, 45, 85, 44, 92, 73, 77, 64,  
 39  
 < 39: 12, 22, 31, 25, 33, 37, 19, 28, 18, 23, 28, 35

32/44

**計算時間の解析**

15個のデータのソート: 42回の比較で十分  
 全体では,  $42 \times (n/15)$  回の比較

$M$ は中央値の中央値であるから,  
 $M$ 以上の中央値をもつグループは  $(n/15)/2$ グループ  
 それらのグループでは半数(8個)以上が中央値以上。  
 よって,  $M$ 以上の要素数は少なくとも  $(8/30)n = (4/15)n$   
 つまり,  $M$ 以下の要素数も  $M$ 以上の要素数も高々  $(11/15)n$ 個

$M$ に関する分割に  $n$ 回の比較が必要

以上より,  
 $T(n) \leq 42(n/15) + T(n/15) + n + T((11/15)n)$   
 よって,  
 $T(n) \leq 19n$ .

練習問題: 上の漸化式を解け.

33/44

**Analysis of Computation Time**

Sort of 15 data: 42 comparisons suffice  
 in total,  $42 \times (n/15)$  comparisons

$M$  is the median of the group medians, and so  
 there are  $(n/15)/2$  groups whose median are  $\geq M$ ,  
 where 8 or more elements (more than half) are  $\geq M$ .  
 Thus, there are at least  $(8/30)n = (4/15)n$  data  $\geq M$ .  
 That is, there are at most  $(11/15)n$  elements  $\leq M$  and  $\geq M$ .

For the decomposition w.r.t.  $M$ ,  $n$  comparisons are required.

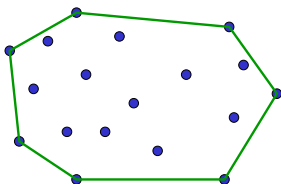
From the above arguments, we have  
 $T(n) \leq 42(n/15) + T(n/15) + n + T((11/15)n)$   
 Hence,  
 $T(n) \leq 19n$ .

Exercise: Solve the above recurrence equation.

34/44

**問題P10: (凸包の構成)**

平面上に与えられた  $n$  点を包含する最小の凸多角形(凸包)を求めよ.

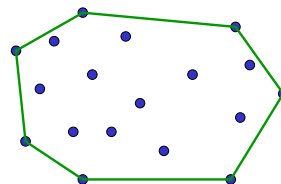


凸包の頂点は必ず与えられた点になる.

35/44

**Problem P10: (Construction of Convex Hull)**

Given a set  $S$  of  $n$  points in the plane, construct a smallest convex polygon (convex hull) containing all the points in its interior.



Vertices of a convex hull must be given points.

36/44

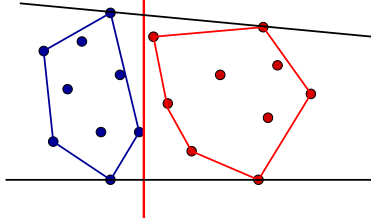
### 凸包の構成

Algorithm P10-A0:

**分割:** 与えられた点を  $x$  座標の中央値で左右に2分割

**統治:** 左右の点集合に対する凸包を再帰的に構成.

**統合:** 2つの凸包の共通外接線を求めて, 2つの凸包を1つの凸多角形に統合する.



37/44

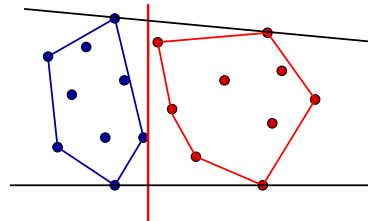
### Construction of Convex Hull

Algorithm P10-A0:

1. **Divide** a set of points into two subsets by a **vertical line** at the median  $x$ -coordinate of the points.

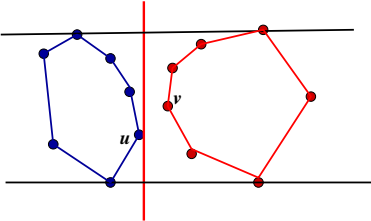
2. Compute the convex hull for each subset **recursively**.

3. **Merge** the two convex hulls into one convex polygon. (Find two external tangent lines to merge them.)



38/44

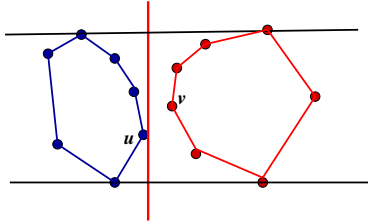
### 2つの凸包の統合



左の凸包の最も右の頂点  $u$  と右の凸包の最も左の頂点  $v$  を見つける.

39/44

### Merging Two Convex Hulls



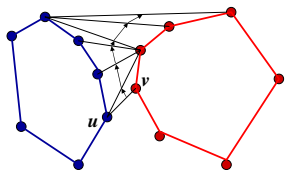
Find the rightmost vertex  $u$  of the left polygon and the leftmost vertex  $v$  of the right polygon.

40/44

ペア  $(u, v)$  から始める

上部接線になるまで, 両方の凸包上を上へ辿る

下部接線になるまで, 両方の凸包上を下へ辿る



毎回どちらかの頂点が次の場所に移動するから, これに要する時間は線形時間.

$(u, v)$  が  $u$  からの上部接線になるのは  $w$  を時計回りで  $v$  の次の頂点とするとき  $(u, v, w)$  が時計回りの順のとき.

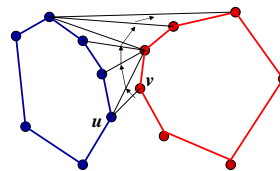
$(u, v)$  が  $v$  からの下部接線になるのは  $t$  を反時計回りで  $u$  の次の頂点とするとき  $(t, u, v)$  が反時計回りの順のとき.

41/44

Starting from the pair  $(u, v)$ ,

go up the pair until it reaches the upper tangent line, and

go down the pair until it reaches the lower tangent line.



It takes linear time since at least one vertex advances to the next position.

$(u, v)$  is an upper tangent line from  $u$  if  $(u, v, w)$  is clockwise order for  $w$ : the next vertex of  $v$  in the clockwise order

$(u, v)$  is an upper tangent line from  $v$  if  $(t, u, v)$  is clockwise order

for  $t$ : the next vertex of  $v$  in the counter-clockwise order

練習問題: 分割統治法に基づく凸包構成アルゴリズムの計算時間は  $O(n \log n)$  であることを示せ.

43/44

Exercise: Show that the algorithm for constructing a convex hull based divide and conquer is  $O(n \log n)$ .

44/44