アルゴリズムとデータ構造

第11週目

担当　情報システム部門　徳光政弘
2025年9月24日

今日の内容

- クイックソートの計算量

クイックソートの手続き

アルゴリズム 6.1　クイックソート

入力：サイズ n の配列 D[0], D[1], ..., D[n-1]

```
quicksort(D,left,right) {
  if (left<right) {
    pivot_index=partition(D,left,right);
    quicksort(D,left,pivot_index-1);
    quicksort(D,pivot_index+1,right);
  }
}
```

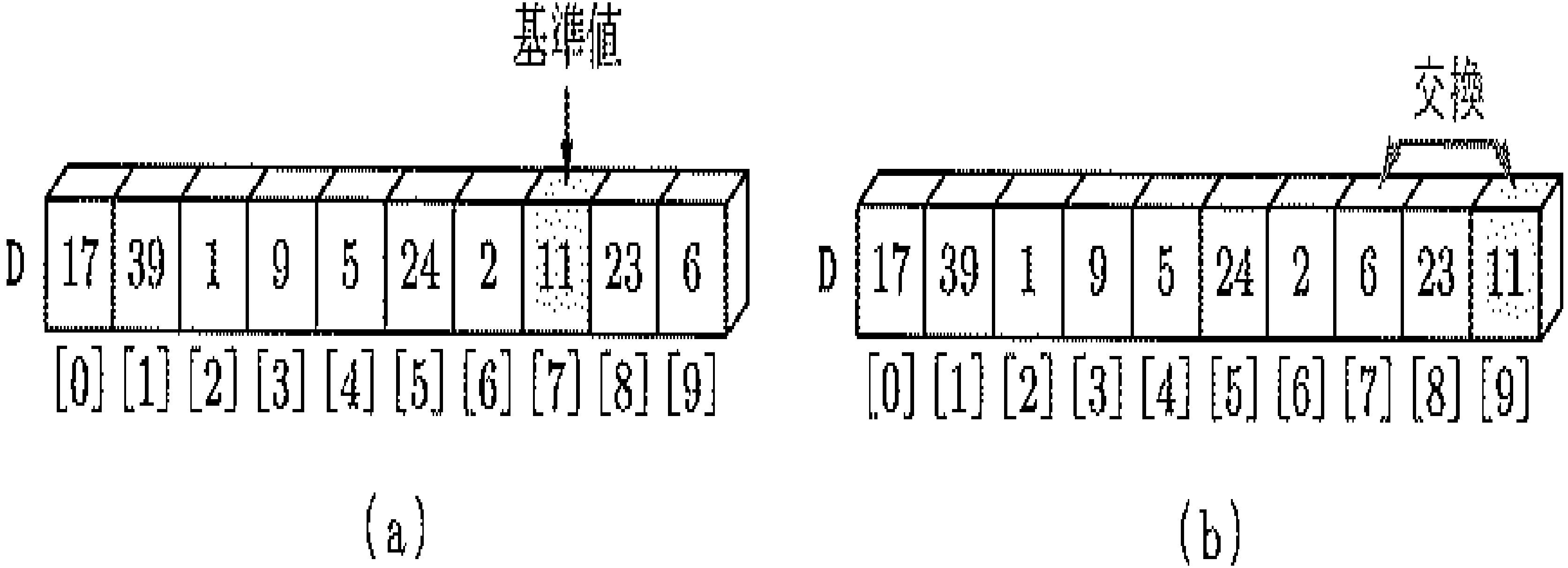
//quicksort(D,0,n-1)|を実行することにより入力全体のソートが実行される．

分割の手順

アルゴリズム 6.2　関数 **partition**

```
partition(D,left,right)  {
  基準値となるデータD[k]を選ぶ;
  swap(D[k],D[right]);        //基準値を右端のデータと交換
  i=left; j=right-1;
  while(i<j)  {
    while (D[i]<D[right]) { i=i+1; }
    while ((D[j]>=D[right])かつ(j>=i)) { j=j-1; }
    if (i<j) swap(D[i],D[j]);
  }
  swap(D[i],D[right]);        //基準値を2つの集合の間に入れる
  return i;                   //基準値の位置を出力
}
```

分割の手順

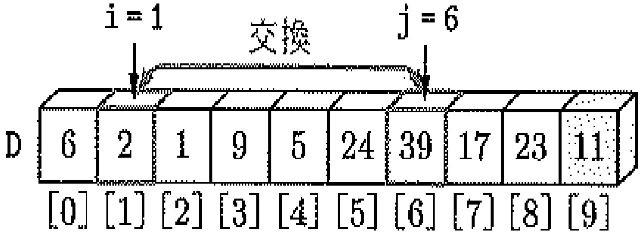


基準値
D
17
39
1
9
5
24
2
11
23
6
[0] [1] [2] [3] [4] [5] [6] [7] [8] [9]
(a)
交換
D
17
39
1
9
5
24
2
6
23
11
[0] [1] [2] [3] [4] [5] [6] [7] [8] [9]
(b)

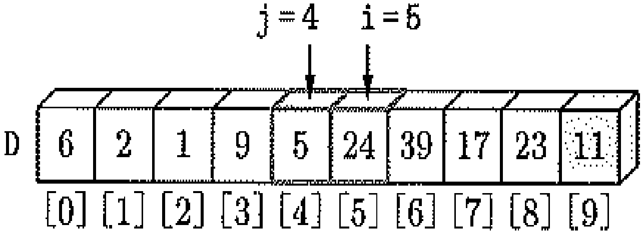
分割の手順

(e)

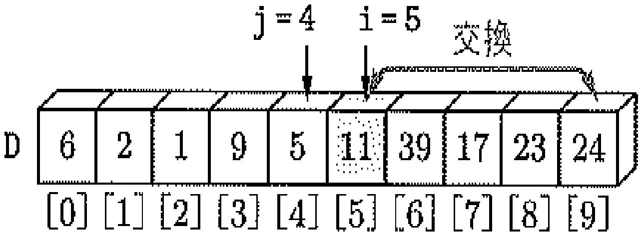
(f)


i=1
交換
j=6
D
6 2 1 9 5 24 39 17 23 11
[0] [1] [2] [3] [4] [5] [6] [7] [8] [9]

(g)


j=4
i=5
D
6 2 1 9 5 24 39 17 23 11
[0] [1] [2] [3] [4] [5] [6] [7] [8] [9]

(h)


j=4
i=5
交換
D
6 2 1 9 5 11 39 17 23 24
[0] [1] [2] [3] [4] [5] [6] [7] [8] [9]

(i)

考え方

- partitionとquicksortの計算量を合わせて考える

① 関数 partition

② 基準値より左の部分の再帰的なクイックソート

③ 基準値より右の部分の再帰的なクイックソート

$$T(n) = \underbrace{cn}_{(1)\ 関数\ \mathtt{partition}} + \underbrace{T(n_{\mathrm{l}})}_{(2)\ 左の部分の再帰} + \underbrace{T(n_{\mathrm{r}})}_{(3)\ 右の部分の再帰} \qquad (n_{\mathrm{l}} + n_{\mathrm{r}} = n - 1)$$

クイックソートの再帰木

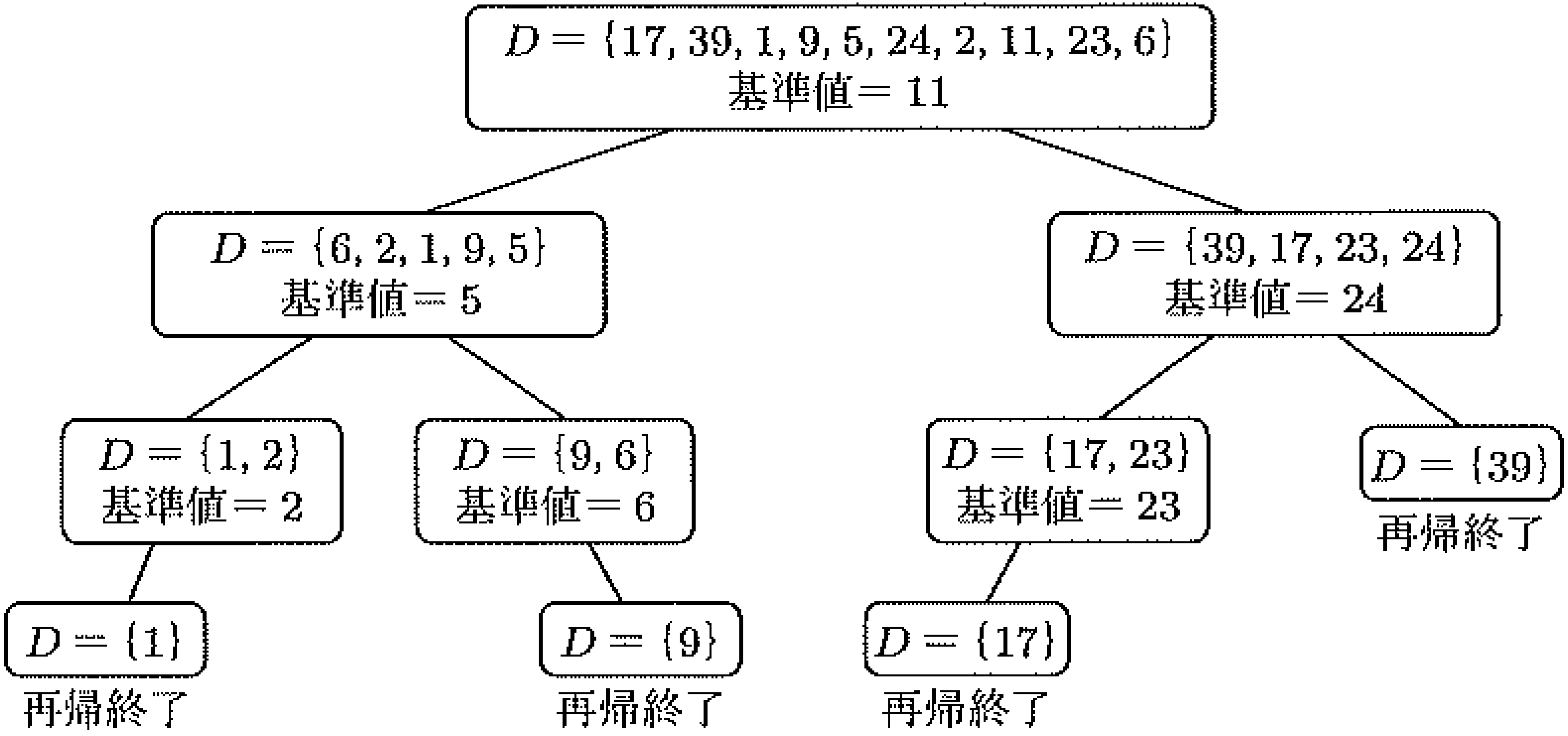


図 6.2　クイックソートの再帰木

考え方

- partitionとquicksortの計算量を合わせて考える

① 関数 partition

② 基準値より左の部分の再帰的なクイックソート

③ 基準値より右の部分の再帰的なクイックソート

$$T(n) = \underbrace{cn}_{(1)\ \text{関数 partition}} + \underbrace{T(n_\mathrm{l})}_{(2)\ \text{左の部分の再帰}} + \underbrace{T(n_\mathrm{r})}_{(3)\ \text{右の部分の再帰}} \qquad (n_\mathrm{l} + n_\mathrm{r} = n - 1)$$

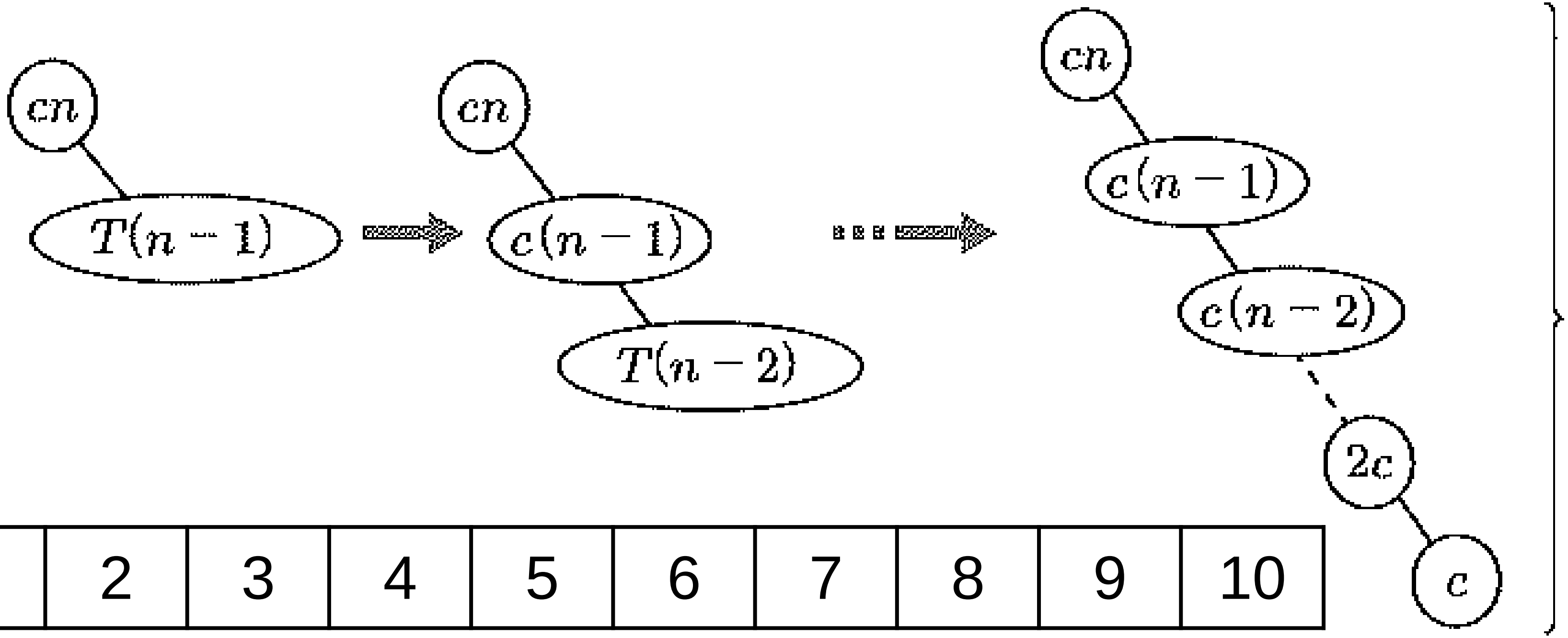
最悪ケースの考え方

- partitionの分割方法で計算量が左右される
- 昇順状態で一番小さい値を使った極端な分割

1	2	3	4	5	6	7	8	9	10

分割点

最悪ケースの場合のソート



1	2	3	4	5	6	7	8	9	10

ソートするたびに分割点が右へずれる

最悪計算量

$$T(n) = \underbrace{cn}_{\text{関数 partition}} + \underbrace{T(n-1)}_{\text{右の部分の再帰}}$$

右の木を順番に展開・ソートすることで、cnの数列となる
（右の木が展開されるたびに分割対象の個数が減る）

$$\sum_{i=0}^{n-1} c(n-i) = \sum_{i=1}^{n} ci = c\frac{n(n+1)}{2} = O(n^2)$$

最良ケース

- 基準値が均等に分割ができる場合

$$T(n) \leq \underbrace{cn}_{\text{関数 partition}} + \underbrace{T\left(\frac{n}{2}\right)}_{\text{左の部分の再帰}} + \underbrace{T\left(\frac{n}{2}\right)}_{\text{右の部分の再帰}}$$

最良ケース

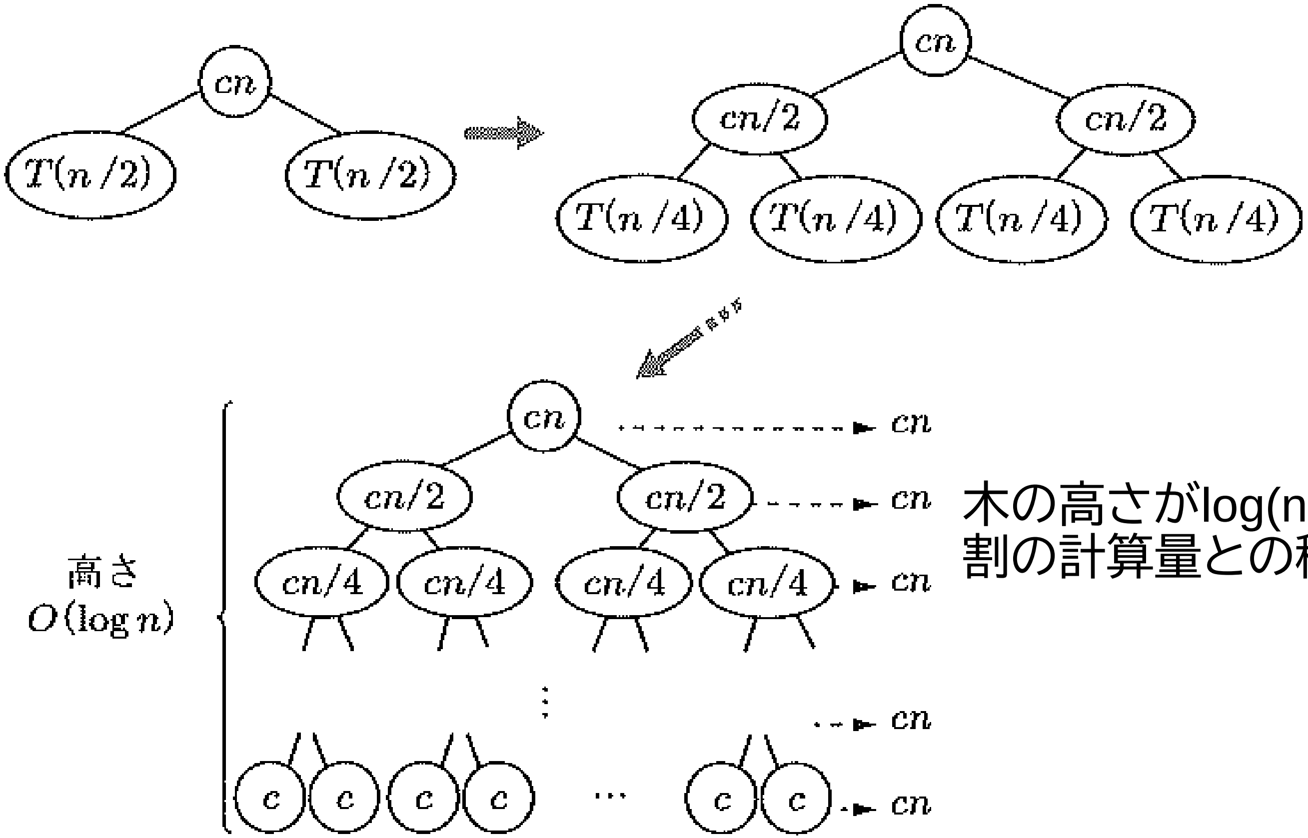


図 6.6　最良時間計算量の場合の再帰木

木の高さがlog(n)になるため、分割の計算量との積となる

平均計算量

- ソート対象の値が確率的に均一に分布していることを仮定する
- 直感では最良ケースのソート(前ページ)を参考にする

●性質 6.1

n 個のデータに対するクイックソートの平均時間計算量は，$O(n \log n)$ である．