計算機ソフトウェア
第三回

電気電子工学科
黒橋禎夫

整列（ソート, sort）

- 順序関係が定義できるデータをその順序関係に従って並べ替える

  3, 7, 1, 9 → 1, 3, 7, 9

- 直感的には$O(n^2)$
- 工夫すると$O(n \log n)$のアルゴリズムがある
  - びっくりするくらい速い

様々なソートアルゴリズム

- バブルソート $O(n^2)$
- 挿入ソート $O(n^2)$
- シェルソート $O(n^2)$
  - 最悪n^2なんですけど、実際もっと速い
- クイックソート $O(n\log n)$
- ヒープソート $O(n\log n)$

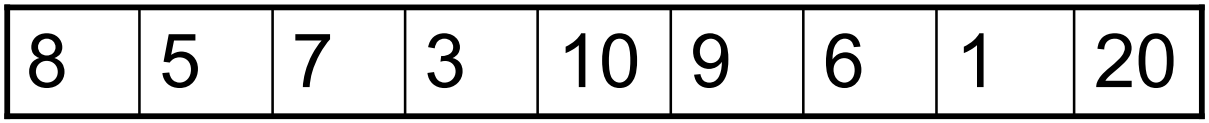
アルゴリズムのセンス

最大のものをみつけるとかはローカルで頑張ってもあまり意味がない

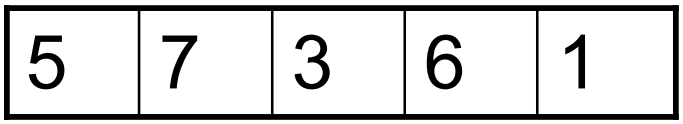
もっとおおらかにやる！

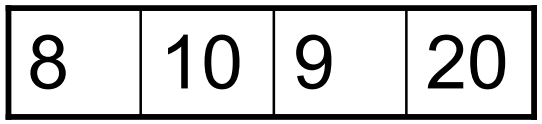
「どっちかというと大きい方を選ぶ」くらいがよい

クイックソート

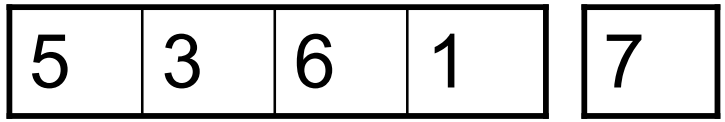


8を基準に大小に分割

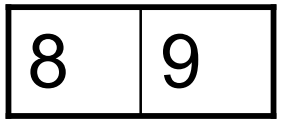


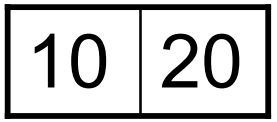


7を基準

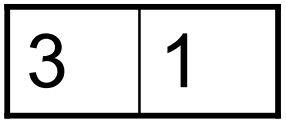


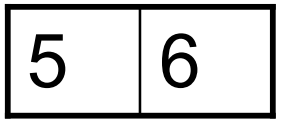
10を基準





5を基準

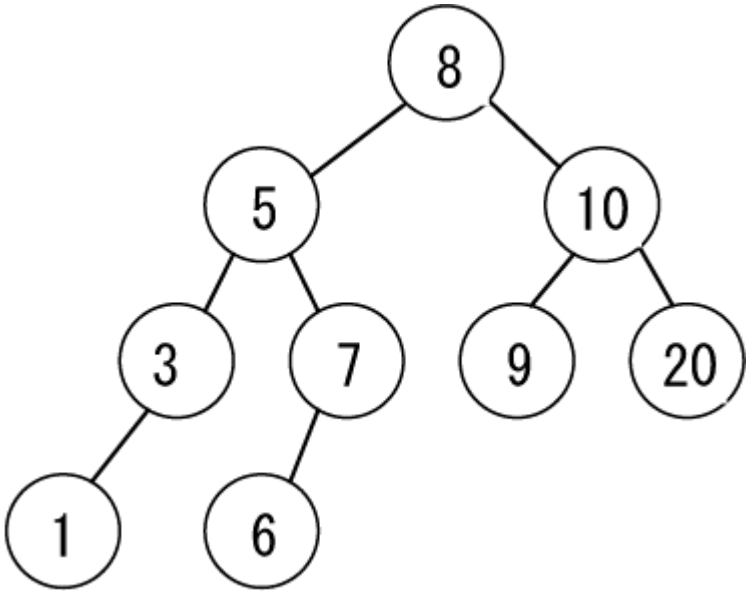




平均O(n*log*n)

最悪$O(n^2)$

２進木(binary tree)を用いたソート



各ノードは高々２つの子をもつ

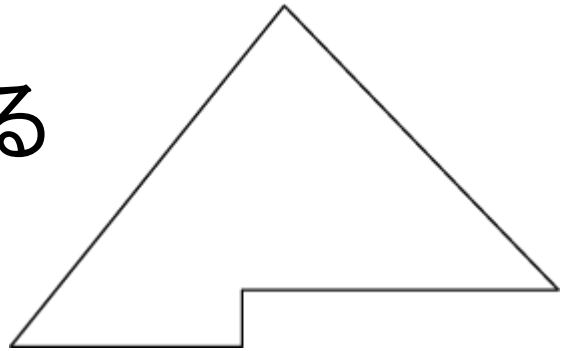
自分より小さければ左の子、大きければ右の子にする

左の子、自分、右の子を順に再帰的に読み出すとソートになる

平均O(n*log*n)　本質的にクイックソートと全く同じアルゴリズム

ヒープ（heap）

- 特別な形の2進木
- k-1段目まで完全に埋める
- K段目は左から隙間無く埋める
- 親子間の関係のみ定める



f(親) ≧ f(子)

２進木によるソートで、左は全部小さくて右は全部大きいというのも実はやり過ぎ

ここでも、頑張りすぎないことが大事です