MEDIAN DAN STATISTIK URUTAN

KULIAH ANALISIS ALGORITMA DAN KOMPLEKSITAS

OUTLINE

- **Masalah pemilihan (selection problem)**
- **Menentukan minimum dan maksimum**
- **Masalah pemilihan dalam expected linear time**
- **Masalah pemilihan dengan worst-case linear time**

ORDER STATISTICS (STATISTIK URUTAN)

- **Statistik urutan ke-i dari himpunan n elemen adalah *elemen terkecil ke-i*.**
- **Minimum: statistik urutan pertama (*i* =1).**
- **Maksimum: statistik urutan ke-n/terakhir (*i* =n).**
- **Median adalah nilai tengah dari himpunan tersebut.**
  - Jika n ganjil: median ada satu, yaitu statistik urutan ke-$(n + 1)/2$.
  - Jika *n* genap, terdapat 2 median:
    - Median bawah: $i = \lfloor (n + 1)/2 \rfloor$
    - Median atas: $i = \lceil (n + 1)/2 \rceil$
    - Dalam kuliah ini: Median → median bawah

MASALAH PEMILIHAN (SELECTION PROBLEM)

- **Adalah masalah yang berkaitan dengan cara menentukan statistik urutan ke-i (dan running time-nya).**
- **Selection problem:**
  - Input: himpunan A dengan n angka, dan bilangan bulat i dengan $1 \leq i \leq n$.
  - Output: elemen $x \in A$ yang lebih besar dari $i - 1$ elemen lain di A (statistik urutan ke-i dari A)
- **Selection problem dapat diselesaikan dalam $O(n \lg n)$ → caranya?**
- **Apakah terdapat algoritma yang lebih cepat?**

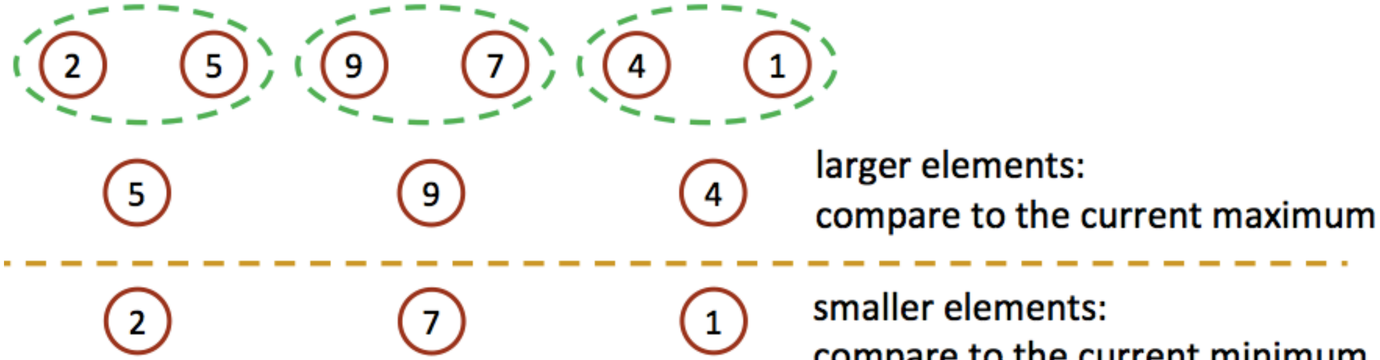
MENENTUKAN MINIMUM

- **Caranya:**
  - Cek setiap elemen dan setiap saat bandingkan dengan elemen terkecil saat itu.
  - Akan terdapat n-1 perbandingan.
  - Algoritma optimal karena setiap elemen harus dibandingkan dengan elemen minimum saat itu.
- **Algoritma:**
  - MINIMUM(*A*)
  1. min = A[1]
  2. for i = 2 to A.length
  3.   if *min* > *A*[*i*] then *min* ← *A*[*i*]
  4. return *min*
- **Maksimum dapat ditentukan dengan cara sama, dengan mengganti > dengan <.**

PENCARIAN MINIMUM DAN MAKSIMUM SECARA SIMULTAN

- **Beberapa aplikasi memerlukan baik minimum maupun maksimum.**
  - Contoh: program grafik yang menskala gambar
- **Cara mudah: tentukan minimum dan maksimum secara independen → memerlukan 2 (n-1) = 2n – 2 perbandingan.**
- **Dapat dilakukan dengan maksimum 3 $\lfloor n/2 \rfloor$ perbandingan:**
  - Minimum dan maksimum disimpan setiap saat.
  - Elemen-elemen diproses secara berpasangan.
  - Bandingkan setiap pasangan elemen.
  - Bandingkan elemen terbesar dengan maksimum, bandingkan elemen terkecil dengan minimum.
  - Berarti hanya 3 perbandingan untuk setiap 2 elemen.

- **Observasi:**
  - Jika pasangan elemen dibandingkan, yang lebih besar tidak mungkin menjadi minimum, yang lebih kecil tidak mungkin menjadi maksimum.
  - Jadi kita hanya perlu membandingkan yang besar dengan maksimum saat ini dan yang kecil dengan minimum saat ini.
  - Biaya: 3 perbandingan untuk setiap 2 elemen (metode sebelumnya: 2 perbandingan untuk setiap elemen).



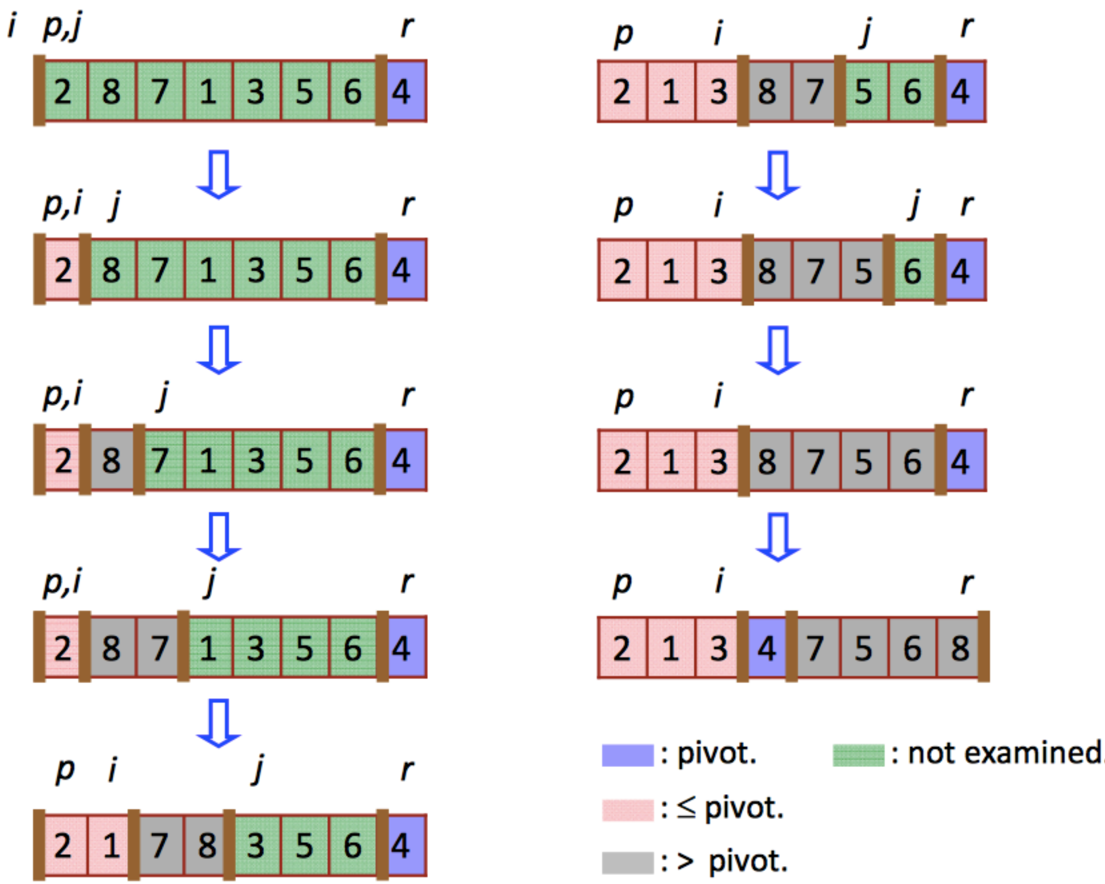
- **Bagaimana menentukan nilai awal min dan max? → tergantung n.**
  - Jika *n* genap, bandingkan 2 elemen pertama, yang besar menjadi max, yang kecil menjadi min.
  - Jika n ganjil, elemen pertama di-set sebagai min dan max.
- **Jika *n* genap, jumlah perbandingan = $3(n - 2)/2 + 1 = 3n/2 - 2$.**
- **Jika n ganjil, jumlah perbandingan = $3(n - 1)/2 = 3 \lfloor n/2 \rfloor$**
- **Untuk kedua kasus: jumlah perbandingan ≤ $3 \lfloor n/2 \rfloor$.**

SELECTION DALAM EXPECTED LINEAR TIME

- **Seleksi terhadap elemen terkecil ke-i dari larik A dapat dilakukan dalam waktu $\Theta(n)$.**
- **Terdapat 2 versi algoritma:**
  - Randomized
  - Deterministik
- **Algoritma RANDOMIZED-SELECT:**
  - Merupakan algoritma divide and conquer.
  - Menggunakan RANDOMIZED-PARTITION pada quicksort.
  - Rekursi hanya pada salah satu sisi partisi.

RECALL: PROSEDUR PARTISI UNTUK QUICKSORT

- **Pada quicksort: larik A[*p* . . *r*] dipartisi menjadi 2 array yang tidak kosong, yaitu *A*[*p* . . *q*] dan *A*[*q*+1 . . *r*], sedemikian hingga setiap elemen pada *A*[*p* . . *q*] kurang dari atau sama dengan elemen-elemen pada *A*[*q*+1 . . *r*].**
- **Prosedur biasa:**
  - Partition (*A*, *p*, *r*)
    - $x \leftarrow A[r]$
    - $i \leftarrow p-1$
    - For j = p to r -1
      - If A[j] $\leq$ x
        - *i = i + 1*
        - *Tukar A[i] dengan A[j]*
    - Tukar A[i + 1] dengan A[r]
    - Return *i + 1*



i p,j r
2 8 7 1 3 5 6 4
p,i j r
2 8 7 1 3 5 6 4
p,i j r
2 8 7 1 3 5 6 4
p,i j r
2 8 7 1 3 5 6 4
p i j r
2 1 7 8 3 5 6 4
p i j r
2 1 3 8 7 5 6 4
p i j r
2 1 3 8 7 5 6 4
p i r
2 1 3 8 7 5 6 4
p i r
2 1 3 4 7 5 6 8
: pivot.
: not examined.
: ≤ pivot.
: > pivot.

PARTISI VERSI RANDOMIZED

- **Tidak menggunakan A[r] sebagai pivot.**
- **Dipilih elemen secara random dari larik/sub-larik yang sedang diurutkan.**
- **RANDOMIZED-PARTITION(*A*, *p*, *r*)**

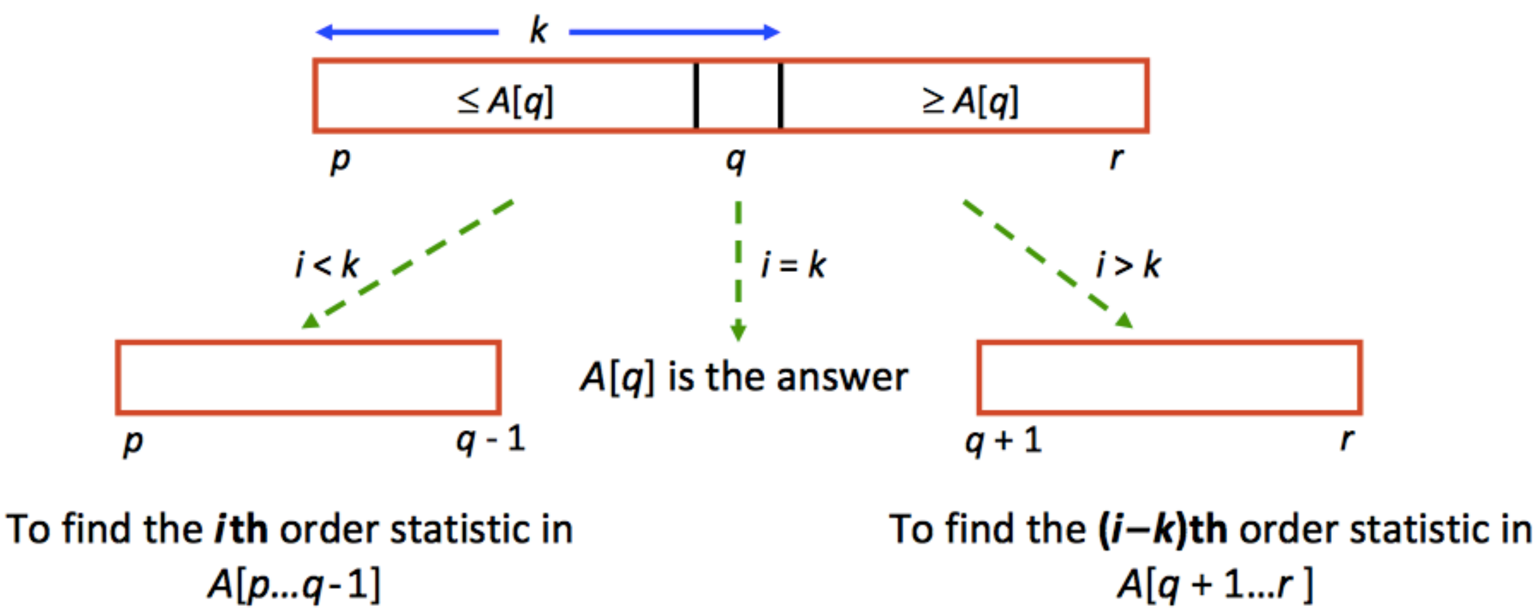
1. *i* ← RANDOM(*p*, *r*)
2. Tukar *A*[*r*] ↔ *A*[*i*]
3. return PARTITION(*A*, *p*, *r*)

PROSEDUR RANDOMIZED SELECT

- **RANDOMIZED-SELECT(*A*, *p*, *r*, *i*)**

1. if $p = r$
2. then return $A[p]$
3. $q \leftarrow$ RANDOMIZED-PARTITION(A, p, r)
4. $k \leftarrow q - p + 1$
5. if $i = k$ /* nilai pivot adalah jawabannya */
6. then return $A[q]$
7. else if $i<k$
8. then return RANDOMIZED-SELECT(A, p, $q - 1$, i)
9. else return RANDOMIZED-SELECT(A, q, r, $i - k$)

- **Visualisasi randomized-select:**



ANALISIS ALGORITMA

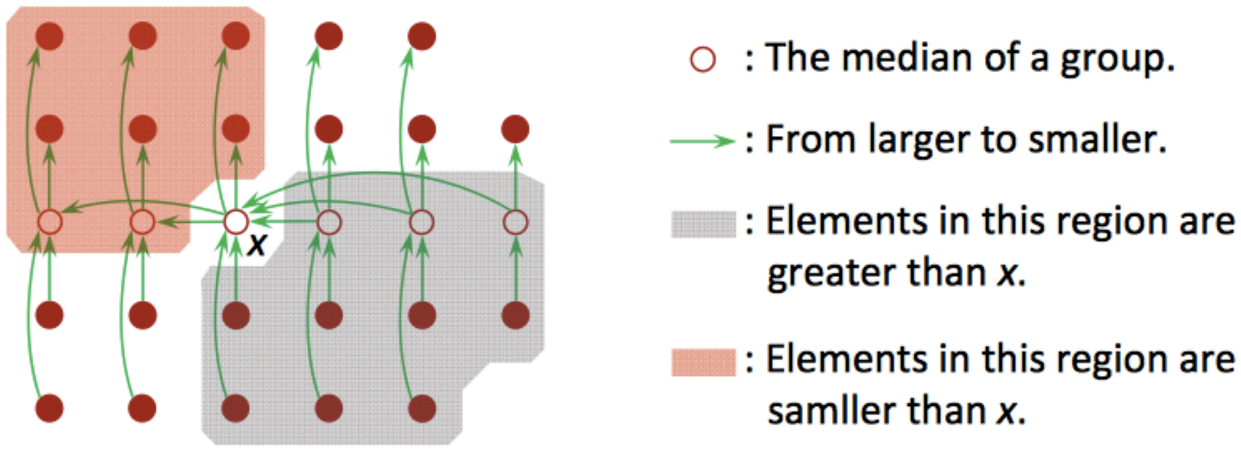
- **Worst case: selalu rekursi pada sub-array yang hanya 1 elemen lebih sedikit dari pada array sebelumnya.**
  - $T(n)=T(n-1)+\Theta(n) = \Theta(n^2)$
- **Best case: selalu rekursi pada sub-array yang hanya memiliki ½ jumlah elemen lebih sedikit dari pada array sebelumnya.**
  - $T(n) = T(n/2\ ) + \Theta(n)$
    $= \Theta(n)$ (Master Theorem, kasus 3)
- **Average case: O(n)**

ALGORITMA SELECT (DETERMINISTIK)

- **Running time O(n) pada worst-case.**
- **Menggunakan rekursi untuk mempartisi elemen input.**
  - Tetapi dipastikan didapatkan pembagian yang bagus.
- **Menggunakan modifikasi dari algoritma PARTITION.**
- **Input: array dari n bilangan, i**
- **Output: elemen terkecil ke-i**

- **Langkah-langkah algoritma SELECT :**
  - Bagi *n* element menjadi kelompok-kelompok, masing-masing terdiri dari 5 elemen, sisanya dimasukkan sebagai grup terakhir.
  - Tentukan median dari masing-masing $\lceil n/5 \rceil$ grup.
    - Lakukan insertion sort untuk setiap grup.
    - Tentukan median dari setiap grup.
  - Gunakan SELECT secara rekursif untuk menentukan median x dari $\lceil n/5 \rceil$ median yang ada.
    - Dalam hal ini, x adalah median dari median.
    - Jika ada sejumlah genap median, berarti x adalah lower median.
  - Partisi *n* elemen dengan pivot x.
    - Misalkan x adalah elemen ke-k dari array setelah partisi.
    - Terdapat k-1 elemen di sebelah kiri dan n-k elemen di sebelah kanan.

- Maka terdapat 3 kemungkinan:
  - Jika i = k, return x.
  - Jika *i* < *k*, gunakan SELECT secara rekursif untuk menentukan elemen terkecil ke-i di sebelah kiri.
  - If *i* > *k*, gunakan SELECT secara rekursif untuk menentukan elemen terkecil ke-(*i*-*k*) di sebelah kanan.



KOMPLEKSITAS

- **Minimal setengah dari median-median tersebut ≥ x → *lebih tepatnya* $\lceil \lceil n/5 \rceil / 2 \rceil$**
- **Setiap grup dengan median ≥ x memuat 3 elemen > x, kecuali:**
  - Grup yang memuat x
  - Grup dengan <5 elemen.
- **Jumlah minimal elemen yang lebih besar dari x :**
  - $3 (\lceil \lceil n/5 \rceil / 2 \rceil - 2) \geq$ **$3n/10 - 6$**.
- **Dengan cara sama: minimal $3n/10 - 6$ elemen yang kurang dari x.**
- **Jadi SELECT dipanggil secara rekursif ≤ $3n/10 - 6$ pada langkah 5.**

- **Kompleksitas setiap langkah:**
  - Bagi *n* elemen menjadi kelompok-kelompok yang masing-masing terdiri dari 5 elemen → O(n)
  - Tentukan median dari masing-masing $\lceil n/5 \rceil$ grup → O(n)
  - Gunakan SELECT untuk menentukan median x dari $\lceil n/5 \rceil$ median yang ada → $T(\lceil n/5 \rceil)$
  - Partisi *n* elemen dengan pivot *x* → O(n)
  - Terdapat 3 kemungkinan → → $T(3n/10 - 6)$
- **Kompleksitas: $T(n) \leq T(\lceil n/5 \rceil) + T(3n/10 - 6) + O(n)$**
- **Untuk T(n) ≤ *cn* untuk *c* besar.**
  - Jika dihitung, kompleksitasnya: O(n)

LATIHAN

- **Dengan data 3, 2, 9, 0, 7, 5, 4, 8, 6, 1 gunakan algoritma untuk mencari minimum dan maksimum secara simultan dengan maksimum 3 $\lfloor n/2 \rfloor$ jumlah perbandingan. Tulislah hasil setiap langkahnya.**
- **Terdapat data: 5, 4, 9, 18, 29, 6, 12, 11, 14, 16, 15, 2, 1, 19, 30, 24, 21. Gunakan algoritma SELECT untuk menentukan elemen terkecil ke-15. Tulislah hasil setiap langkahnya.**
- **Dari data: 3, 2, 9, 0, 7, 5, 4, 8, 6, 1, gunakan algoritma RANDOMIZED-SELECT untuk menentukan elemen terkecil ke-8. Bilangan random pada RANDOMIZED-PARTITION dapat Anda pilih sendiri. Tulis hasil setiap langkahnya.**