Randomized Algorithm

William Gozali
Pelatnas 3 TOKI 2015

Peringatan!

- Slide ini mengabaikan aspek perhitungan kompleksitas secara formal, jadi jangan mengacu pada slide ini untuk perkuliahan tentang algoritma atau statistika
- Namun, slide ini tetap dirancang supaya ide dan penjelasan kenapa randomized algorithm bagus dapat dimengerti

Strategi Randomized...

- Las Vegas
- Monte Carlo

Las Vegas

- Memanfaatkan keteracakan untuk mempercepat eksekusi algoritma
- Algoritma tetap menghasilkan **jawaban yang benar**
- Contoh: randomized quicksort, order statistic, treap

Randomized Quicksort

- Pilih elemen yang akan menjadi pivot secara acak
- Algoritma tetap memiliki worst case $O(N^2)$
- Secara rata-rata bekerja dalam O(N log N), biasa disebut sebagai expected O(N log N)
- Apa yang dimaksud expected?

Expected Number

- Terjemahan ke Bahasa Indonesia: nilai harapan
- Contoh:

  Peluang hari hujan adalah 0.1

  Jika Pelatnas berlangsung 21 hari, kira-kira berapa hari yang diharapkan terjadi hujan?

  = 2.1 hari

Expected Number (lanj.)

- Sebuah dadu cacat akan menghasilkan suatu angka dengan peluang berikut:

Angka	Peluang
1	0.3
2	0.1
3	0.2
4	0.1
5	0.2
6	0.1

- Jika dadu dilempar 1x, berapa expected angka yang muncul?

Expected Number (lanj.)

Angka	Peluang
1	0.3
2	0.1
3	0.2
4	0.1
5	0.2
6	0.1

= 1*0.3 + 2*0.1 + 3*0.2 + 4*0.1 + 5*0.2 + 6*1

= 3.1

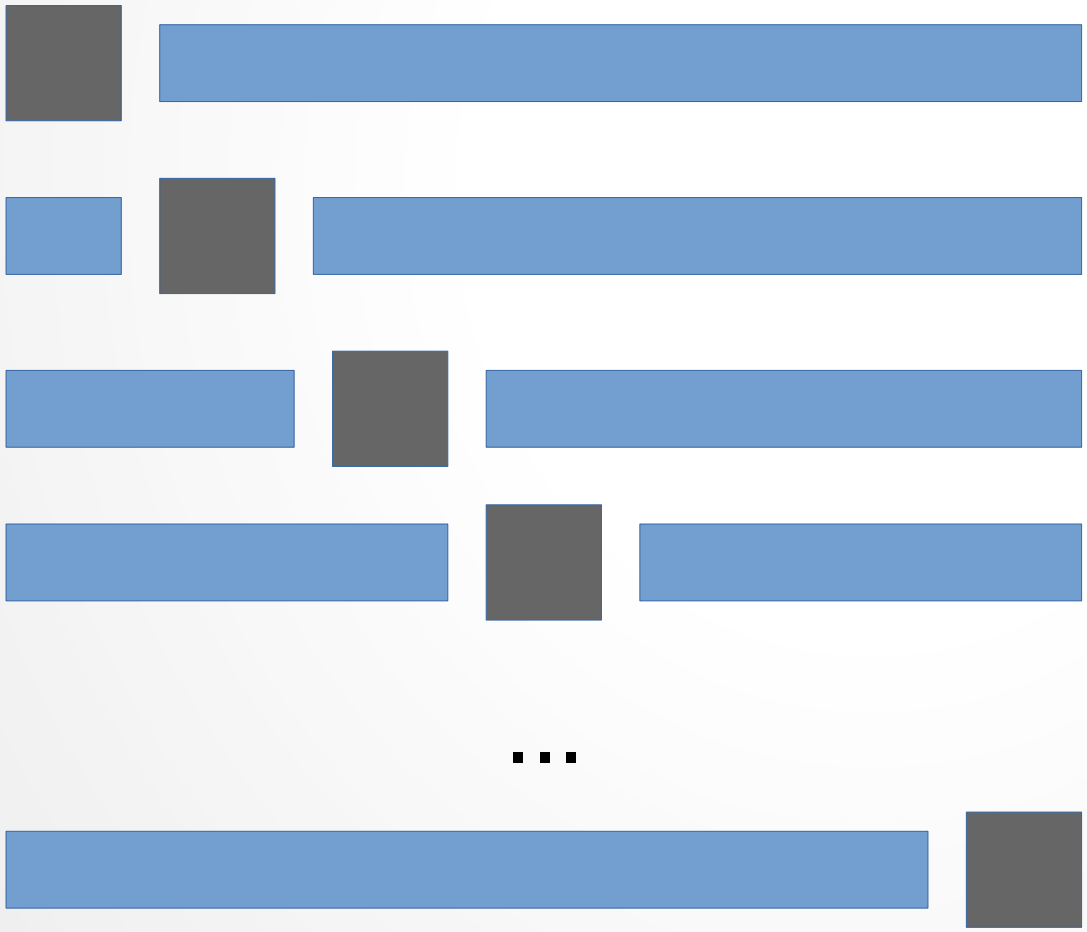
- Cara menghitungnya adalah menjumlahkan semua nilai angka*peluang

Randomized Quicksort (lanj.)

- Misalkan angka yang akan diurutkan ada pada subarray A[l..r]
- Asumsikan semua angka unik
- Jika digunakan random pivot, setiap elemen A[i] memiliki peluang yang sama untuk menjadi pivot

Randomized Quicksort (lanj.)

- Semua kemungkinan partisi yang mungkin terjadi adalah:



Randomized Quicksort (lanj.)

- Ada r-l+1 kemungkinan partisi, masing-masing peluang terjadinya adalah 1/(r-l+1)

Randomized Quicksort (lanj.)

- Jika T(n) menyatakan kompleksitas expected randomized quick sort untuk n data, maka :

$$T(n) = \frac{1}{n} \sum_{i=1}^{n} \left(T(i-1) + T(n-i-1) \right) + O(n)$$

- Penyelesaian untuk hubungan T(n) tersebut adalah T(n) = O(n log n)
- Jadi kompleksitas expected dari randomized quick sort adalah O(n log n)

Order Statistic

- Diberikan array berisi N bilangan, cari bilangan terkecil ke-K
- Solusinya mirip seperti pada partisi quick sort:
  - Pilih salah satu elemen secara acak, pindah semua elemen yang lebih kecil dari elemen tersebut ke kiri dengan algoritma partisi (seperti pada quicksort)
  - Jika banyaknya elemen di sebelah kiri == K-1, artinya bilangan yang dipilih ini adalah bilangan ke-K
  - Jika banyaknya elemen di sebelah kiri < K-1, artinya jawabannya ada di kanan. Ulangi prosedur ini untuk subarray di kanan
  - Jika banyaknya elemen di sebelah kiri > K-1, artinya jawabannya ada di kiri. Ulangi prosedur ini untuk subarray di kiri

Order Statistic (lanj.)

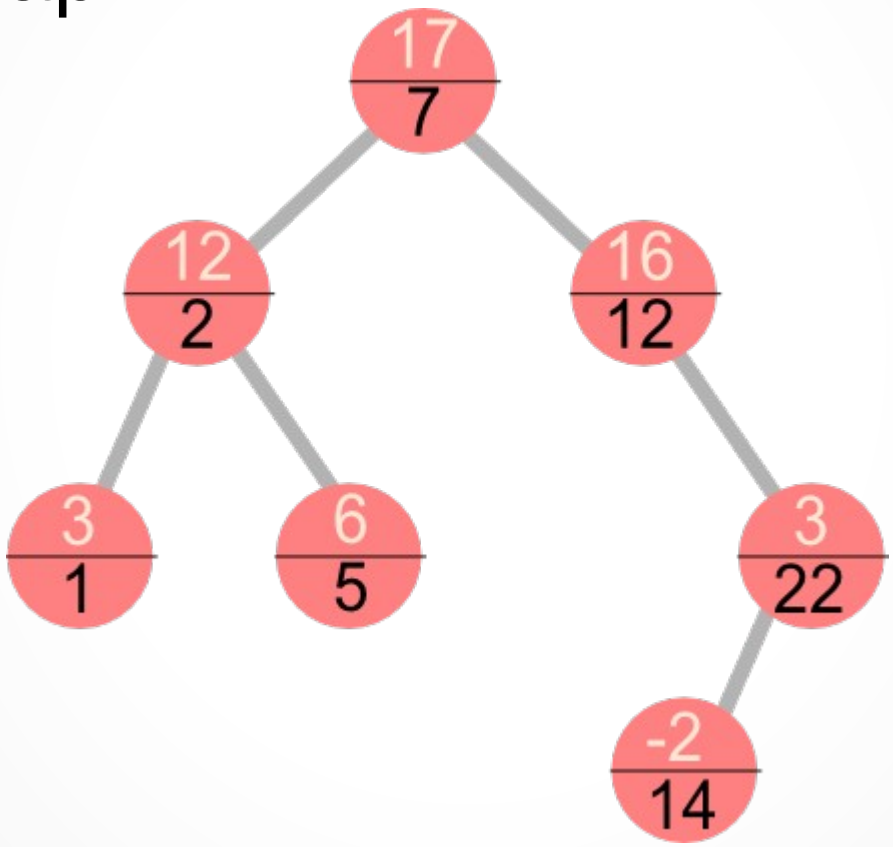
- Array = [1, 5, 3, 2, 4], cari bilangan terkecil ke-2
- Pilih salah satu elemen, misalnya 3
- Setelah dipartisi:
  - Bagian kiri = [1, 2]
  - Bagian kanan = [5, 4]
- Jadi solusinya ada di bagian kiri, ulangi lagi sampai ketemu
- Kompleksitas expected-nya adalah O(N)

Treap

- Varian dari BBST
- Setiap node menyimpan informasi tambahan: prioritas
- Nilai prioritas ini di-random
- Untuk setiap node, nilai prioritasnya harus lebih besar dari prioritas anak-cucunya (seperti heap)
- Untuk setiap node, nilai key-nya harus lebih besar dari seluruh key anak-cucu di subtree kiri, dan lebih kecil dari seluruh key anak-cucu di subtree kanan (seperti BST)

Treap (lanj.)

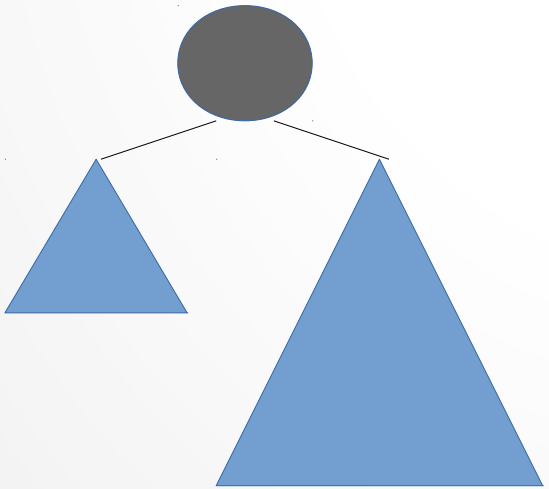
- Contoh treap



- Berapa kedalaman expected dari treap?

Treap (lanj.)

- Analisisnya mirip seperti randomized quicksort
- Setiap elemen memiliki peluang yang sama untuk menjadi root node



- Kedalaman expected-nya adalah O(log N)

Rangkuman

- Semua algoritma dan struktur data itu menggunakan pendekatan Las Vegas
- Sifat keteracakan dimanfaatkan untuk membuat algoritma lebih cepat, jarang terjebak di worst-case
- Jawaban dari algoritma/proses struktur data selalu benar

Monte Carlo

- Strategi “coba-coba”
- Bentuk umumnya:

```
jawaban = null
while (belumPuas(jawaban)){
   tebakJawaban = random();
   if (lebihBaik(tebakJawaban, jawaban)){
      jawaban = tebakJawaban;
   }
}
```

Monte Carlo (lanj.)

- Bisa juga untuk melakukan pemeriksaan yang sulit/lambat, tetapi diketahui prosedur cepat untuk memberikan informasi:
  - Jawaban mungkin benar
  - Jawaban pasti salah
- Contoh: periksa apakah suatu bilangan prima dan diketahui prosedur cepat untuk memberikan informasi:
  - Bilangan mungkin prima
  - Bilangan pasti bukan prima
- Pemeriksaan dengan cara ini diimplementasikan pada algoritma Miller-Rabin primality testing

Monte Carlo (lanj.)

- Hasilnya **belum tentu benar**, meskipun mungkin **hampir benar**
- Baik digunakan jika jawaban yang dicari tidak harus paling optimal, cukup “hampir optimal”

Aproksimasi Luas Objek

- Hitung luas buah pada gambar berikut:

Aproksimasi Luas Objek (lanj.)

- Cara mudah: loop untuk setiap pixel-nya, increment luas bila warna pixel itu kekuningan, semacam flood-fill
- Kekurangannya: banyak pixel yang harus diperiksa
- Bagaimana jika:
  - Algoritma harus dijalankan pada mesin yang kecepatannya terbatas? Contoh: microcontroller robot
  - Jawaban tidak harus eksak

Aproksimasi Luas Objek (lanj.)

- Strategi Monte Carlo:
  - Pilih beberapa pixel saja, misalnya N
  - Misalnya di antara N pixel itu, ada K pixel yang warnanya kekuningan
  - Aproksimasi luas buah = K/N * luas gambar
- Semakin banyak N (titik sampel), semakin akurat hasil perhitungan

Verifikasi Perkalian Matriks

- Diberikan 3 matriks berukuran N*N, misalnya bernama A, B, dan C
- Dinyatakan bahwa A*B = C
- Periksa apakah benar A*B = C!

Verifikasi Perkalian Matriks (lanj.)

- Jika N cukup besar (misalnya 2000), mengalikan matriks N*N bisa lambat
- Strategi Monte Carlo:
  - Pilih elemen pada matriks C, misalnya C[i][j]
  - Periksa apakah A[i][:] * B[:][j] = C[i][j]
  - Ulangi untuk nilai i dan j yang lainnya sampai puas
  - Jika ada satu saja pasangan i dan j yang tidak memenuhi A[i][:] * B[:][j] = C[i][j], artinya A*B bukan C
- Kompleksitasnya O(kN), dengan k = banyak percobaan

Aplikasi Lainnya

- Algoritma primality testing → Miller-Rabin
- Dunia number theory dan analisis probabilitas
- Dunia fisika: fluida, gerakan
- Dunia IOI: soal output only