Solusi UTS 2013

1. **(*Divide and Conquer***) Sebuah bilangan disebut *sederhana* jika ia terdiri dari angka yang berulang, misalnya 3333, 7777, 444444, dan sebagainya.
2. Formulasikan cara mengalikan dua buah bilangan sederhana yang masing-masing panjangnya *n* angka dengan algoritma *divide and conquer*, nyatakan kompleksitas waktunya dalam persamaan rekurens, lalu selesaikan dan perlihatkan bahwa kompleksitas waktunya ternyata *O*(*n*), yang dalam hal ini kompleksitas waktu dihitung berdasarkan jumlah operasi kali dan operasi tambah. Asumsikan *n* adalah perpangkatan dari dua. **(Nilai = 15 + 3 + 2 )**
3. Lakukan perkalian 3333 × 7777 dengan algoritma *divide and conquer* di atas, lalu hitung berapa jumlah operasi kali dan operasi tambahnya (catatan: 10*s* dinyatakan sebagai mengisi 0 sebanyak *s* kali). **(Nilai = 5)**

Jawaban: Misalkan X dan Y adalah integer sepanjang n angka

(a)

 *X a a*

 *Y b b*

 *n*/2 *n*/2

*X* = *a* ⋅ 10n/2 + *a*

*Y* = *b* ⋅ 10*n/2* + *b*

 XY = (*a* ⋅ 10*n/2* +*a*) ⋅ (*b* ⋅ 10*n/2* + *b*)

 = *ab* ⋅ 10n + *ab* ⋅ 10*n/2* + *ab* ⋅ 10*n/2* + *ab*

 = *ab* (10n + 2 ⋅ 10*n/2* + 1) 🡪 hanya dibutuhkan satu kali perkalian untuk n/2 bagian

(Nilai = 15)

  (Nilai = 3)

 T(n) = T(n/2) + q

 a = 1, b = 2, d = 1 🡪 a < bd 🡪 1 < 21 🡪 T(n) = O(n1) = O(n) (Nilai = 2)

1. 3333 = 33⋅102 + 33

 7777 = 77⋅102 + 77

 3333 × 7777 = (33⋅102 + 33)( 77⋅102 + 77)

 = 33⋅77⋅104 + 33⋅77⋅102 + 33⋅77⋅102 + 33⋅77

 33 = 3⋅10 + 3

 77 = 7⋅10 + 7

 33 × 77 = (3⋅10 + 3)( 7⋅10 + 7)

 = 3⋅7⋅102 + 3⋅7⋅10 + 3⋅7⋅10 + 3⋅7

 = 3⋅7 (102 + 10 + 10 + 1)

 Jumlah operasi kali = hanya 1 buah saja (yaitu mengalikan 3 ⋅ 7)

 Jumlah operasi tambah = 6 buah

1. **(*Decrease and Conquer*)**
2. Tuliskan kompleksitas waktu (dalam persamaan rekurens *T*(*n*) = *aT*(*n*/*b*) + *cnd* ) algoritma *binary search* yang mencari sebuah elemen tertentu di dalam tabel terurut yang berukuran *n*. Kemudian, selesaikan relasi rekurens tersebut dan nyatakan kompleksitas waktu asimptotiknya dalam notasi *O*-besar.

**(Nilai = 2 + 5 + 2)**

1. Gunakan Teorema Master untuk menyelesaikan persamaan rekurens tersebut untuk menemukan kompleksitas waktu asimptotik yang sama seperti jawaban (a) Anda di atas.

**(Nilai = 3)**

1. Misalkan algoritma *binary search* dimodifikasi sedemikian sehingga ia membagi tabel menjadi tiga bagian yang berukuran sama (disebut juga algoritma *ternary search*). Tuliskan kompeksitas waktu *ternary search* dalam persaman rekurens. Kemudian, selesaikan relasi rekurens tersebut. Dibandingkan dengan kompleksitas waktu *binary search*, mana yang lebih baik? Selanjutnya, nyatakan kompleksitas waktu asimptotiknya dalam notasi *O*-besar?

**(Nilai = 2 + 3 + 3 + 2)**

1. Gunakan Teorema Master untuk menyelesaikan persamaan rekurens tersebut untuk menemukan kompleksitas waktu asimptotik *ternary search* (hasilnya sama dengan jawaban (c))

 **(Nilai = 3)**

Jawaban:

2. (a) (Nilai = 2)

 T(n) = 1 + T(n/2)

 = 1 + (1 + T(n/4)) = 2 + T(n/4)

 = 2 + (1 + T(n/8)) = 3 + T(n/8)

 = … = j + T(n/2j)

 Asumsi: n = 2j 🡪 j = 2log n

 T(n) = 2log n + T(1) = 2log n + 1= 1 + 2log n = O(2log n) (Nilai = 5 + 2)

(b) *T*(*n*) = *aT*(*n*/*b*) + *cnd*

 T(n) adalah

 T(n) = T(n/2) + 1 = T(n/2) + n0

a = 1, b = 2, d = 0 🡪 a = bd 🡪 1 = 20 🡪 T(n) = O(n0 log n) = O(log n) (Nilai = 3)

(c) (Nilai = 2)

 T(n) = T(n/9) + 2 + 2 = T(n/9) + 4 = T(n/27) + 2 + 4 = T(n/27) + 6 = … = T(n/3k) + 2k

 Berhenti ketika n/3k = 1 🡪 n = 3k 🡪 k = 3log n 🡪 T(n) = T(1) + 2 (3log n) = 1 + 2 3log n

 (Nilai = 3)

 Karena 1 + 2 3log n > 1 + 2log n, maka tetap lebih baik binary search (Nilai = 3)

 T(n) = 1 + 2 3log n = O(log n) (Nilai = 2)

(d) T(n) = T(n/3) + 2 = T(n/3) + 2.n0

a = 1, b = 3, d = 0 🡪 a = bd 🡪 1 = 30 🡪 T(n) = O(n0 log n) = O(log n) (Nilai = 3)

1. **(Algoritma Greedy)** Sebuah supermarket sedang merancang sistem otomatis yang dapat memberikan kenyamanan lebih baik bagi pelanggannya. Pengembang sistem akan memilih apakah akan menerapkan algoritma *brute force* atau *greedy* dalam mengimplementasikan sistem yang diinginkan. Bantulah pengembang sistem dengan menjelaskan strategi serta efektifitas (penjaminan solusi tercapai) dan efisiensi (kompleksitas algoritma) untuk setiap instansiasi persoalan yang diberikan. Berikanlah solusinya.
	1. Fasilitas penentuan lintasan terpendek dari pintu masuk ke setiap produk yang diinginkan seorang pelanggan, yaitu makanan beku, elektronik, sayur & buah, susu, dan mie, dan kembali ke kasir. Posisi setiap produk, jalur yang tersedia, dan panjang jalur digambarkan pada graf berikut ini. Fasilitas ini memberikan urutan produk yang akan diambil, lalu jalur terpendek dari produk yang terakhir diambil ke kasir.

masuk

Makanan beku

Elektronik

Sayur & buah

susu

mie

kasir

3

5

2

5

1

8

10

2

6

3

**(Nilai = 12,5)**

* 1. Fasilitas penentuan seminar promosi produk yang akan diikuti seorang pelanggan. Setiap seminar promosi akan memberikan *cash-back* yang diasumsikan sama besarnya, sehingga setiap pelanggan berusaha mengikuti seminar promosi sebanyak-banyaknya. Misalkan pelanggan membeli 8 produk yang mengadakan seminar promosi dengan informasi sbb:

|  |  |  |
| --- | --- | --- |
| Produk | Waktu mulai | Waktu selesai |
| makanan beku 1 | 1 | 4 |
| makanan beku 2 | 2 | 4 |
| Elektronik 1 | 1 | 3 |
| Elektronik 2 | 5 | 7 |
| sayur & buah | 4 | 7 |
| susu 1 | 3 | 4 |
| Susu 2 | 6 | 8 |
| Mie 1 | 4 | 5 |
| Mie 2 | 7 | 8 |

**(Nilai = 12,5)**

Jawaban:

1. **Gunakan Dijkstra**

**Urutan produk: elektronik, makanan beku, sayur & buah, susu, mie.**

**Jalur terpendek: mie – susu – sayur & buah – makanan beku - kasir**

****

1. **Strategi: durasi promosi terkecil lebih dahulu**

|  |  |  |  |
| --- | --- | --- | --- |
| **Produk** | **Waktu mulai** | **Waktu selesai** | **Durasi** |
| **makanan beku 1** | **1** | **4** | **3** |
| **makanan beku 2** | **2** | **4** | **2** |
| **Elektronik 1** | **1** | **3** | **2** |
| **Elektronik 2** | **5** | **7** | **2** |
| **sayur & buah** | **4** | **7** | **3** |
| **susu 1** | **3** | **4** | **1** |
| **Susu 2** | **6** | **8** | **2** |
| **Mie 1** | **4** | **5** | **1** |
| **Mie 2** | **7** | **8** | **1** |

**Elektronik1 (1,3), Susu 1 (3,4), mie (4,5), elektronik 2 (5,7), mie 2 (7,8)**

1. (DFS dan BFS) Misalkan anda mempunyai dua buah ember kosong *A* dan *B*, masing-masing dapat diisi penuh 5 liter dan 3 liter. Anda diminta mendapatkan air (dari sebuah danau) sebanyak 4 liter di dalam salah satu ember dengan menggunakan bantuan hanya kedua ember tersebut (tidak ada peralatan lain yang tersedia, hanya kedua ember itu saja yang ada!). Anda boleh menggunakan operator berikut ini:

*fill* (*x*) : mengisi ember *x* sampai penuh. Syarat: ember *x* tidak penuh.

*clear* (*x*) : membuang seluruh air dari ember *x*. Syarat: ember *x* tidak kosong.

*move* (*x*,*y*): memindahkan air dari ember *x* ke ember *y* sampai *y* penuh, dan isi ember *x* berkurang. Syarat: ember *x* memiliki isi lebih banyak atau sama dengan *z* liter untuk membuat *y* penuh.

Anda harus menjelaskan apa yang menjadi *state* persoalan. Gambarkan pencarian solusi persoalan ini dengan membangun pohon secara dinamis dengan algoritma *DFS* atau *BFS*. Pohon hanya mengandung *state* yang unik. Urutan operator harus digunakan secara konsisten.

Berikanlah penomoran *state* untuk menggambarkan urutan *state* pada pohon yang dibangun. Urutan operator: *fill*(*A*), *fill*(*B*), *clear*(*A*), *clear* (*B*), *move*(*A*, *B*), *move*(*B*, *A*). Operator yang tidak dipergunakan tidak perlu dicantumkan pada pohon. **(Nilai = 25)**

**Jawaban:**

State: isi air (dalam liter) setiap ember <isi ember A 5 liter, isi ember B 3 liter >.

**Solusi dgn BFS: TBD**

S0 <0,0>:

1-- fill (A): S1 <5,0>

2----- fill (A): -

2----- fill (B): S3 <5,3>

2----- clear (A): -{kembali ke S0}

2----- clear (B): -

2----- move (A,B): S4 <2,3>

3-------- fill (A): - { kembali ke S3}

3-------- fill (B): -

3-------- clear (A): - {kembali ke S2}

3-------- clear (B): S

2----- move (B,A): -

1-- fill (B): S2 <0,3>

2----- fill (A): - {kembali ke S3}

2----- fill (B): -

2----- clear (A): -

2----- clear (B): - {kembali ke S0}

2----- move (A,B): -

2----- move (B,A): S5 <3,0>

1-- clear (A): -

1-- clear (B): -

1-- move (A,B): -

1-- move (B,A): -