MIDTERM DAA (KOMS119602 / KOMS120403)

Tuesday, March 29th 2022 (07.30 - 10.00 WITA)

Petunjuk

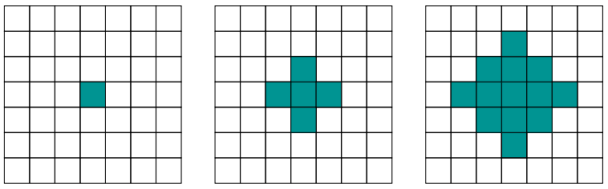
1. Durasi ujian adalah **120 menit** (100 menit pengerjaan soal, dan 20 menit untuk persiapan + submission).
2. Ujian dilaksanakan secara online melalui google meet. Anda diwajibkan terhubung ke google meet selama pelaksanaan ujian.
3. Anda **diharuskan** mengerjakan ujian secara mandiri. Dilarang keras mencontek/berdiskusi dengan siapapun. Ujian ini openbook, namun Anda dilarang mencari materi di Internet.
4. Tulis jawaban dengan singkat (tidak bertele-tele) namun jelas. Tulis dalam Bahasa Indonesia. Hindari menggunakan tinta merah. Kerjakan soal secara berurutan dan tulis semua nomor soal yang dikerjakan meskipun Anda tidak mengerjakannya. Pada setiap lembar jawaban, berikan nomor halaman.

Centang pernyataan berikut pada google form sebelum mengerjakan soal:
"Saya menyatakan bahwa saya mengerjakan UTS ini dengan sejujur-jujurnya, tanpa bantuan orang lain dan tanpa menggunakan cara yang tidak dibenarkan. Apabila di kemudian hari diketahui saya mengerjakan UTS ini dengan cara yang tidak jujur, saya bersedia mendapatkan konsekuensinya, yaitu mendapatkan nilai E pada mata kuliah DAA Ilkom Semester Genap T.A. 2021/2022."

1 Algorithms and complexity analysis

1. (**Kompleksitas algoritma**)

   Diberikan algoritma pewarnaan untuk mewarnai kotak satu satuan sebagai berikut: pada iterasi pertama, dimulai dengan mewarnai satu kotak di tengah dan, pada setiap n iterasinya, menambahkan kotak baru di sekelilingnya. Konfigurasi untuk iterasi 1, 2, dan 3 diilustrasikan pada gambar di bawah ini.



   (a) Berapa banyak kotak ukuran satu satuan yang dihasilkan oleh algoritma tersebut pada iterasi ke-i?

   (b) Nyatakan fungsi kompleksitas waktu algoritma tersebut dengan formula rekursi, kemudian selesaikan formula tersebut untuk mendapatkan fungsi eksplisit kompleksitas waktunya. (*Hint:* Untuk mendapatkan formula rekursif, nyatakan fungsi kompleksitas waktunya dalam suatu formula $f(i) = f(i-1) + (ai + b)$ untuk suatu konstanta a dan b.)

Solution:

(a) Misal $N(i)$: banyaknya persegi setelah iterasi ke-i. Pola peningkatan jumlah kotak dapat diamati secara diagonal. Pada iterasi ke-i, terdapat i pola diagonal yang terdiri dari i kotak, dan $i-1$ pola yang terdiri dari $i-1$. Maka:

$$N(i) = i^2 + (i-1)^2 = 2i^2 - 2i + 1$$

(b) Fungsi kompleksitas waktu juga dapat dinyatakan dengan formula rekursif, yaitu:

$$N(i) = N(i-1) + 4i - 4$$

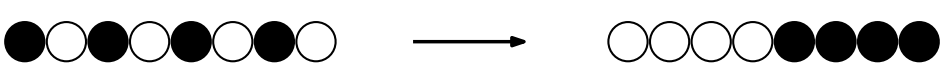
dimana $N(1) = 1$. Banyaknya kotak pada iterasi ke-i diperoleh dengan menambahkan jumlah kotak pada iterasi ke-$i-1$ dan kotak di sekelilingnya (berbentuk belah ketupat) sebanyak $4i-4$.

Dengan menyelesaikan formula rekursif diperoleh:

$$\begin{aligned}
N(i) &= N(i-1) + 4i - 4 \\
&= (N(i-2) + 4(i-1) - 4) + 4i - 4 = N(i-2) + (4i + 4(i-1)) - 4 \cdot 2 \\
&= (N(i-3) + 4(i-2) - 4) + 4(i-1) - 4 = N(i-3) + (4i + 4(i-1) + 4(i-2)) - 4 \cdot 3 \\
&\vdots \\
&= N(1) + (4i + 4(i-1) + 4(i-2) + \cdots + 4 \cdot 2) - 4(i-1) \\
&= 1 + 4 \cdot \frac{1}{2} \cdot (i-1)(i+2) - 4(i-1) \\
&= 2i^2 - 2i + 1
\end{aligned}$$

2. (**Sorting**)

   Diberikan $2n$ bola yang berwarna hitam dan putih masing-masing sebanyak n, dan disusun secara berselang-seling sebagaimana ditunjukkan pada gambar berikut. Anda ditugaskan untuk mengatur susunan bola-bola tersebut sehingga semua bola hitam akan terkumpul di bagian kanan, dan semua bola putih terkumpul di bagian kiri. Pada setiap langkah, Anda hanya diizinkan untuk menukar posisi dua bola yang berurutan.



(a) Desain sebuah algoritma untuk menyelesaikan masalah ini, dan hitung kompleksitas waktunya (berdasarkan pada jumlah perpindahan yang dilakukan).

(b) Apakah algoritma yang Anda desain sudah optimal? Berikan argumen singkat, mengapa?

Solution:

(a) Algoritma:

(a) Misal indeks bola hitam (dari kiri ke kanan) adalah $1, 3, 5, \ldots, 2n-1$*, dan indeks bola putih adalah* $2, 4, 6, \ldots, 2n$*.*

(b) Dimulai dari bola putih pertama (indeks 2), tukar posisinya dengan bola hitam 1. Sehingga hasilnya bola putih 2 ada di posisi ke-1.

(c) Lanjutkan dengan bola pada indeks berikutnya (putih 4), dan secara sequential tukar dengan bola d

(d) Lanjutkan proses untuk semua bola putih, sampai dengan bola putih terakhir (indeks $2n-1$*), yang akan berpindah ke posisi ke-*n*.*

Kompleksitas waktu dihitung dari banyaknya perpindahan yaitu:

$$1 + 2 + 3 + \cdots + n = \frac{n(n+1)}{2} \in \mathcal{O}(n^2)$$

(b) Algoritma tersebut optimal karena ...

3. (**Polynomial interpolation**)

Coba ingat kembali algoritma interpolasi polinomial, untuk mengevaluasi polinomial

$$p(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0$$

di titik $x = t$.

(a) Algoritma tersebut memiliki kompleksitas waktu *worst case* $\mathcal{O}(n^2)$. Desain sebuah algoritma untuk masalah ini dengan kompleksitas waktu $\mathcal{O}(n)$ (i.e. linier). Jelaskan strateginya dan tuliskan pseudocode untuk algoritma tersebut!

(b) Apakah mungkin untuk mendesain algoritma untuk masalah ini dengan kompleksitas waktu yang lebih baik dari $\mathcal{O}(n)$? Berikan argumen Anda!

Solution: *Algoritma yang diberikan di kelas:*

Algorithm 1 Polynomial interpolation

```
1:  procedure POLYNOM(n, A[0..n], t)
2:      p ← 0
3:      for i ← n downto 0 do
4:          power ← 1
5:          for j ← 1 to i do
6:              power ← power * t
7:          end for
8:          p ← p + a[i] * power
9:      end for
10:     return p
11: end procedure
```

(a) Algoritma tersebut tidak efisien karena pada setiap iterasinya, kita harus menghitung perpangkatan dari x. Kita bisa memanfaatkan relasi $x^k = x * x^{k-1}$. Jadi untuk meningkatkan efisiensi, kita dapat menghitung perpangkatan yang berurutan (misal x^k dan x^{k-1}) dengan lebih efisien.

Kita dapat menghitung dari suku tertinggi ke terendah, atau menghitung x^{k-1} dari x^k. Tetapi pendekatan ini membutuhkan akan membutuhkan pembagian, dan dengan demikian kita akan membutuhkan perlakuan khusus ketika $x = 0$.

Atau kita dapat menghitung mulai dari pangkat terendah ke tertinggi, yaitu menghitung x^k dari x^{k-1}. Pseudocode dapat ditulis sebagai berikut:

Algorithm 2 Polynomial interpolation

```
1: procedure POLYNOM(n, A[0..n], t)
2:     p ← 0; power ← 1
3:     for i ← 1 to n do
4:         power ← power * t
5:         p ← p + a[i] * power
6:     end for
7:     return p
8: end procedure
```

Banyaknya operasi perkalian adalah:

$$M(n) = \sum_{i=1}^{n} 2 = 2n$$

dan banyaknya operasi penjumlahan adalah n. Sehingga kompleksitas waktunya adalah ($\mathcal{O}(n)$).

(b) Tidak, karena algoritma apa pun untuk mengevaluasi suatu polinomial dengan derajat n pada suatu titik $x = t$ harus memproses semua koefisien $n + 1$. (Perhatikan bahwa meskipun $x = 1$, $p(x) = a_n + a_{n-1} + \cdots + a_1 + a_0$, membutuhkan setidaknya n operasi penjumlahan.

2 Polynomials multiplication

In this exercise, we investigate a divide-and-conquer approach to multiply two polynomials of the same order n (similar to the "Matrix multiplication" and "Large numbers multiplication" discussed in the lecture).

1. (**Naive polynomial multiplication algorithm**)
   Given two polynomials of the same order n as follows. Our goal is to compute $A(x)B(x)$.

$$A(x) = a_0 + a_1x + a_2x^2 + \cdots + a_nx^n$$
$$B(x) = b_0 + b_1x + b_2x^2 + \cdots + b_nx^n$$

   A naive way to perform the polynomials multiplication is by *direct multiplication*, as in the following example:
   Example

$$A(x) = 1 + 2x + 3x^2$$
$$B(x) = 3 + 2x + 2x^2$$

$A(x)B(x) = (1 + 2x + 3x^2)(3 + 2x + 2x^2) = 3 + 8x + 15x^2 + 10x^3 + 6x^4$ □

(a) Solve the following polynomials multiplication using naive algorithm!

$$A(x) = 2 + 5x + 3x^2 + x^3 - x^4$$
$$B(x) = 1 + 2x + 2x^2 + 3x^3 + 6x^4$$

(b) Design a brute-force algorithm to multiply two polynomials $A(x)$ and $B(x)$ that are of order n.

(c) Compute the complexity of your brute-force algorithm. Represent it using an asymptotic notation!

Solution:

(a)

$$A(x) \cdot B(x) = 2 + 9x + 17x2 + 23x3 + 34x^4 + 39x^5 + 19x^6 + 3x^7 - 6x^8$$

(b) Algoritma paling sederhana adalah dengan mengalikan setiap suku pada $A(x)$ dengan setiap suku $B(x)$.

(a) Dimulai dari $i = 0$, kalikan $A(x)$ dengan $b_i x^i$

(b) Jumlahkan suku-suku pada penjumlahan dengan pangkat x yang sama.

Untuk setiap iterasi (pada langkah 1), kita dapat membuat array dengan ukuran $n + 1 + i$.

(c) Kompleksitas waktu:

- *Terdapat $n + 1$ iterasi, dimana pada setiap iterasi terdapat $n + 1$ operasi perkalian skalar dan n penjumlahan.*
- *Di akhir, kita menjumlahkan $n + 1$ polinom berderajat $n, n + 1, \ldots 2n$ (yang dihasilkan pada setiap iterasi).*
- *Jadi banyaknya operasi adalah $\mathcal{O}(n^2)$.*

2. (**Polynomials multiplication divide-and-conquer algorithm**)

Given two polynomials of the same order n as follows. Our goal is to compute $A(x)B(x)$.

$$A(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n$$
$$B(x) = b_0 + b_1 x + b_2 x^2 + \cdots + b_n x^n$$

How do we perform polynomials multiplication by Didive-and-Conquer? The algorithm is as follows.

- Split $A(x)$ into $A_0(x)$ and $A_1(x)$, each contains $n/2$ terms:

$$A_0(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_{\lceil n/2 \rceil - 1} x^{\lceil n/2 \rceil - 1}$$
$$A_1(x) = a_{\lceil n/2 \rceil} + a_{\lceil n/2 \rceil + 1} x + a_{\lceil n/2 \rceil + 2} x^2 + \cdots + a_{n - \lceil n/2 \rceil} x^{n - \lceil n/2 \rceil}$$

  So that

$$A(x) = A_0(x) + A_1(x) x^{\lceil n/2 \rceil}$$

- Similarly, $B(x)$ can be split into $B_0(x)$ and $B_1(x)$, so that:

$$B(x) = B_0(x) + B_1(x) x^{\lceil n/2 \rceil}$$

Hence:

$$A(x)B(x) = A_0(x)B_0(x) + \left(A_0(x)B_1(x) + A_1(x)B_0(x)\right) x^{\lceil n/2 \rceil} + A_1(x)B_1(x) x^{2\lceil n/2 \rceil}$$

Task: Solve the following polynomials multiplication using the algorithm explained above. Write the steps clearly!

$$A(x) = 2 + 5x + 3x^2 + x^3 - x^4$$
$$B(x) = 1 + 2x + 2x^2 + 3x^3 + 6x^4$$

Solution:

Note that $n = 4$, and $\lceil n/2 \rceil = 2$.

- $A_0(x) = 2 + 5x$
- $A_1(x) = 3 + x - x^2$
- $B_0(x) = 1 + 2x$
- $B_1(x) = 2 + 3x + 6x^2$
- $A_0(x)B_0(x) = 2 + 9x + 10x^2$
- $A_0(x)B_1(x) = 4 + 16x + 27x^2 + 30x^3$
- $A_1(x)B_0(x) = 3 + 7x + x^2 - 2x^3$
- $A_1(x)B_1(x) = 6 + 11x + 19x^2 + 3x^3 - 6x^4$

Hence:

$$\begin{aligned} A(x)B(x) &= (2 + 9x + 10x^2) + \left((4 + 16x + 27x^2 + 30x^3) + (3 + 7x + x^2 - 2x^3)\right) x^2 + (6 + 11x + 19x^2 + 3x^3 - 6x^4)x^4 \\ &= (2 + 9x + 10x^2) + (7 + 23x + 28x^2 + 28x^3)x^2 + (6 + 11x + 19x^2 + 3x^3 - 6x^4)x^4 \\ &= (2 + 9x + 10x^2) + (7x^2 + 23x^3 + 28x^4 + 28x^5) + (6x^4 + 11x^5 + 19x^6 + 3x^7 - 6x^8) \\ &= 2 + 9x + 17x^2 + 23x^3 + 34x^4 + 39x^5 + 19x^6 + 3x^7 - 6x^8 \end{aligned}$$

3. (**DnC-based polynomials multiplication: pseudocode and time complexity**)
   The divide-and-conquer algorithm explained in question 3 can be written in a pseudocode as follows:

Algorithm 3 Polynomials multiplication (divide-and-conquer, version 1)

```
1:  procedure PolyMul(A, B: polynomials, n: integer)
2:      declaration
3:          A_0, A_1, B_0, B_1: polynomials
4:          s: integer
5:      end declaration
6:      if n = 0 then   return A * B                                   ▷ scalar multiplication
7:      else
8:          s ← ⌈n/2⌉
9:          A_0 ← a_0 + a_1 x + a_2 x^2 + ··· + a_{s-1} x^{s-1}
10:         A_1 ← a_s x^s + a_{s+1} x^{s+1} + a_{s+2} x^{s+2} + ··· + a_n x^{n-s}
11:         B_0 ← b_0 + b_1 x + b_2 x^2 + ··· + b_{s-1} x^{s-1}
12:         B_1 ← b_s x^s + b_{s+1} x^{s+1} + b_{s+2} x^{s+2} + ··· + b_n x^{n-s}
13:         return PolyMul(A_0, B_0, s) + PolyMul(A_0, B_1, s) + PolyMul(A_1, B_0, s) *x^s + PolyMul(A_1, B_1, s) *x^{2s}
14:     end if
15: end procedure
```

Task: Write the time complexity function of the divide-and-conquer algorithm above in a recursive formula. Using Master theorem, compute the asymptotic time complexity!

Solution:

There are 4 recursive calls of problem size $n/2$, and the additions take $\mathcal{O}(n)$-time or cn-time for some constant c. So:

$$T(n) = \begin{cases} t, & \text{for } n = 0 \\ 4T(n/2) + cn, & \text{for } n > 0 \end{cases}$$

By Master theorem, $a = 4$, $b = 2$, and $d = 1$. So, $a > b^d$ (or $4 > 2^1$). The 3rd case of Master theorem is satisfied, so:

$$T(n) \in \mathcal{O}(n^{\log_2 4}) = \mathcal{O}(n^2)$$

4. **(DnC-based polynomials multiplication: improvement)**
Now we want to modify the divide-and-conquer algorithm for the polynomials multiplication given in question 2. We will reduce the number of multiplications performed in the algorithm.
In question 2, we have:

$$A(x)B(x) = A_0(x)B_0(x) + (A_0(x)B_1(x) + A_1(x)B_0(x))\, x^{\lceil n/2 \rceil} + A_1(x)B_1(x)x^{2\lceil n/2 \rceil}$$

There are 4 multiplications and 3 additions of polynomials of order n. We will reduce the number of multiplications to 3, but with a consequence that the number of additions is increased.
Define:

$$\begin{aligned} Y(x) &= (A_0(x) + A_1(x)) \times (B_0(x) + B_1(x)) \\ U(x) &= A_0(x)B_0(x) \\ Z(x) &= A_1(x)B_1(x) \end{aligned}$$

Then:

$$Y(x) - U(x) - Z(x) = A_0(x)B_1(x) + A_1(x)B_0(x)$$

so that:

$$\begin{aligned} A(x)B(x) &= A_0(x)B_0(x) + (A_0(x)B_1(x) + A_1(x)B_0(x))x^{\lceil n/2 \rceil} + A_1(x)B_1(x)x^{2\lceil n/2 \rceil} \\ &= U(x) + (Y(x) - U(x) - Z(x))\, x^{\lceil n/2 \rceil} + Z(x)x^{2\lceil n/2 \rceil} \end{aligned}$$

Note that in this algorithm, there are only three multiplications, namely $U(x)$, $Y(x)$, and $Z(x)$.

Task: Solve the following polynomials multiplication using the algorithm explained above. Write the steps clearly!

$$\begin{aligned} A(x) &= 2 + 5x + 3x^2 + x^3 - x^4 \\ B(x) &= 1 + 2x + 2x^2 + 3x^3 + 6x^4 \end{aligned}$$

Solution: *From the previous question, we obtain:*

- $A_0(x) = 2 + 5x$
- $A_1(x) = 3 + x - x^2$
- $B_0(x) = 1 + 2x$
- $B_1(x) = 2 + 3x + 6x^2$
- $A_0(x)B_0(x) = 2 + 9x + 10x^2$
- $A_1(x)B_1(x) = 6 + 11x + 19x^2 + 3x^3 - 6x^4$

So:

- $Y(x) = \big((2 + 5x) + (3 + x - x^2)\big) \times (1 + 2x) + (2 + 3x + 6x^2) = (5 + 6x - x^2) \times (3 + 5x + 6x^2) = 15 + 43x + 57x^2 + 31x^3 - 6x^4$
- $U(x) = A_0(x)B_0(x) = 2 + 9x + 10x^2$
- $Z(x) = A_1(x)B_1(x) = 6 + 11x + 19x^2 + 3x^3 - 6x^4$

- $Y(x) - U(x) - Z(x) = (15 + 43x + 57x^2 + 31x^3 - 6x^4) - (2 + 9x + 10x^2) - (6 + 11x + 19x^2 + 3x^3 - 6x^4) = 7 + 23x + 28x^2 + 28x^3$

Hence:

$$\begin{aligned} A(x)B(x) &= U(x) + (Y(x)U(x)Z(x))x^2 + Z(x)x^4 \\ &= (2 + 9x + 10x^2) + (7 + 23x + 28x^2 + 28x^3)x^2 + (6 + 11x + 19x^2 + 3x^3 - 6x^4)x^4 \\ &= (2 + 9x + 10x^2) + (7x^2 + 23x^3 + 28x^4 + 28x^5) + (6x^4 + 11x^5 + 19x^6 + 3x^7 - 6x^8) \\ &= 2 + 9x + 17x^2 + 23x^3 + 34x^4 + 39x^5 + 19x^6 + 3x^7 - 6x^8 \end{aligned}$$

5. (**DnC poly-multiplication improvement: pseudocode**)

   The divide-and-conquer algorithm explained in question 3 can be written in a pseudocode as follows:

Algorithm 4 Polynomials multiplication (divide-and-conquer, version 2)

```
1:  procedure PolyMul2(A, B: polynomials, n: integer)
2:      declaration
3:          A_0, A_1, B_0, B_1, U, Y, Z: polynomials
4:          s: integer
5:      end declaration
6:      if n = 0 then
7:          return A * B                                   ▷ scalar multiplication
8:      else
9:          s ← ⌈n/2⌉
10:         A_0 ← a_0 + a_1 x + a_2 x^2 + ··· + a_{s-1} x^{s-1}
11:         A_1 ← a_s x^s + a_{s+1} x^{s+1} + a_{s+2} x^{s+2} + ··· + a_n x^{n-s}
12:         B_0 ← b_0 + b_1 x + b_2 x^2 + ··· + b_{s-1} x^{s-1}
13:         B_1 ← b_s x^s + b_{s+1} x^{s+1} + b_{s+2} x^{s+2} + ··· + b_n x^{n-s}
14:         Y ← PolyMul2(A_0 + A_1, B_0 + B_1, s)
15:         U ← PolyMul2(A_0, B_0, s)
16:         Z ← PolyMul2(A_1, B_1, s)
17:         return U + (Y − U − Z) * x^s + Z * z^{2s}
18:     end if
19: end procedure
```

Task: Write the time complexity function of the divide-and-conquer algorithm above in a recursive formula. Using Master Theorem, compute the asymptotic time complexity! What do you observe?

Solution: *There are 3 recursive calls of problem size $n/2$, and the additions take $\mathcal{O}(n)$-time or cn-time for some constant c. So:*

$$T(n) = \begin{cases} t, & \text{for } n = 0 \\ 3T(n/2) + cn, & \text{for } n > 0 \end{cases}$$

By Master theorem, $a = 3$, $b = 2$, and $d = 1$. So, $a > b^d$ (or $3 > 2^1$). The 3rd case of Master theorem is satisfied, so:

$$T(n) \in \mathcal{O}(n^{\log_2 3}) = \mathcal{O}(n^{1.59})$$

This gives a more efficient algorithm for polynomials multiplication.

6. (**Analysis**)

   After computing the result of $A(x)B(x)$, check if the three different algorithms (brute-force, divide-and-conquer, and the improved version of divide-and-conquer) above give the same result. If not, explain why!

Solution: *You should obtain the same results from the three algorithms above. If not, that means that there exists a mistake somewhere in your computation.*