UNIVERSITI SAINS MALAYSIA

Second Semester Examination
2015/2016 Academic Session

June 2016

CCS592 – Advanced Algorithms and Complexity
[Algoritma Lanjutan & Kekompleksan]

Duration : 2 hours
[Masa : 2 jam]

INSTRUCTIONS TO CANDIDATE:
[ARAHAN KEPADA CALON:]

- Please ensure that this examination paper contains **FOUR** questions in **TEN** printed pages before you begin the examination.

  *[Sila pastikan bahawa kertas peperiksaan ini mengandungi **EMPAT** soalan di dalam **SEPULUH** muka surat yang bercetak sebelum anda memulakan peperiksaan ini.]*

- Answer **ALL** questions.

  *[Jawab **SEMUA** soalan.]*

- You may answer the questions either in English or in Bahasa Malaysia.

  [Anda dibenarkan menjawab soalan sama ada dalam bahasa Inggeris atau bahasa Malaysia.]

- In the event of any discrepancies, the English version shall be used.

  [Sekiranya terdapat sebarang percanggahan pada soalan peperiksaan, versi bahasa Inggeris hendaklah diguna pakai.]

1. (a) Given the time complexity of a sorting algorithm X is c_1n^2 and another sorting algorithm Y is $c_2n\log_{10}n$. Assuming that the processor on computer A executes 10^9 instructions/sec and computer B executes 10^7 instructions/sec. If c_1 is 5 and c_2 is 10, estimate the time to sort 10^6 integers on computers A and B using both algorithms.

Diberi kekompleksan masa algoritma pengisihan X ialah c_1n^2 dan algoritma pengisihan Y ialah $c_2n\log_{10}n$, Andaikan bahawa pemproses pada komputer A melaksanakan 10^9 arahan/saat dan komputer B melaksanakan 10^7 arahan/saat. Jika c_1 ialah 5 dan c_2 ialah 10, anggarkan masa yang diperlukan untuk mengisih 10^6 integer pada komputer A dan B menggunakan kedua-dua algoritma tersebut.

(20/100)

(b) The polynomial p(x) is calculated as follow:

Polinomial p(x) dikira seperti yang berikut:

$$p(x) = \sum_{i=0}^{n} a_i x^i \quad \text{or /atau} \qquad \text{(Eq. 1)}$$

$$p(x) = a_0 + a_1x^1 + a_2x^2 + \cdots + a_nx^n \qquad \text{(Eq. 2)}$$

Horner's rule calculates p(x) in a more efficient manner. Given $x=x_0$, Horner's rule evaluates $p(x_0)$ as follows:

Peraturan Horner mengira p(x) dengan lebih cekap. Diberi $x=x_o$, peraturan Horner menilai $p(x_0)$ seperti yang berikut:

$$b_n = a_n$$
$$b_{n-1} = a_{n-1} + b_n x_0$$
$$b_{n-2} = a_{n-2} + b_{n-1} x_0$$
$$\ldots$$
$$b_1 = a_1 + b_2 x_0$$
$$b_0 = a_0 + b_1 x_0$$

b_0 is the value of $p(x_0)$

b_0 ialah nilai bagi $p(x_0)$

(i) If you write a simple algorithm to calculate p(x) directly based on the formula above (Eq. 2), what is the time complexity of the algorithm? Explain your answer.

Jika anda menulis sebuah algoritma untuk mengira p(x) secara terus dengan rumus di atas (Eq. 2), apakah kekompleksan masa untuk algoritma tersebut? Terangkan jawapan anda.

(ii) What is the complexity of Horner's rule in big-Oh notation? Explain your answer.

Apakah kekompleksan peraturan Horner dalam notasi Oh besar? Terangkan jawapan anda.

(iii) Convert Horner's rule to a Java code. Assume a_i and x are integer.

Tukar peraturan Horner kepada sebuah kod Java. Anggap a_i dan x ialah integer.

(80/100)

2. (a) A server has m processors. When someone submits a job with n instructions into the server, the scheduler will complete the job in the shortest time. The scheduler task is to schedule the execution of the instruction in one of the processor in sequence. The instructions will first be read by one of the processor, and then each instruction can be executed in the processor. Time taken to complete an instruction may be different in different processor. After an instruction is completed, the next instruction can be executed in the same processor or transferred to another processor for processing. Transferring instructions from one processor to another processor will take up some time. Assume the instructions cannot be executed in parallel.

Sebuah pelayan mempunyai m pemproses. Apabila seseorang memberikan satu tugas dengan n arahan kepada pelayan, penjadual akan menyiapkan tugas berkenaan dalam masa yang paling singkat. Tugas penjadual adalah untuk menjadualkan pelaksanaan arahan dalam sesebuah pemproses dalam urutan. Arahan akan dibaca oleh salah sebuah pemproses pada permulaan, dan kemudian setiap arahan boleh dilaksanakan dalam pemproses berkenaan. Masa yang diambil untuk melaksanakan sesuatu arahan mungkin berbeza dalam pemproses yang berbeza. Selepas sesuatu arahan selesai, arahan seterusnya boleh dilaksanakan dalam pemproses yang sama atau dipindahkan ke pemproses lain untuk pemprosesan. Memindahkan arahan daripada suatu pemproses kepada pemproses lain akan mengambil sedikit masa. Andaikan arahan tidak boleh dilaksanakan secara selari.

(i) What is the strategy that can be applied on the scheduler for scheduling the instruction efficiently? Describe the algorithm.

Apakah strategi yang boleh dilaksanakan pada penjadual untuk menjadualkan arahan dengan cekap? Terangkan algoritma tersebut.

(ii) If Billy submits a job with 4 instructions (v, w, x, y) into the server with 3 processors (A, B and C). How the scheduler will schedule the instructions so that they are completed in the shortest time, and how long to complete the job given the information below?

Jika Billy memasukkan sebuah tugas dengan 4 arahan (v, w, x, y) ke dalam pelayan dengan 3 pemproses (A, B dan C), Bagaimanakah penjadual akan menjadualkan arahan supaya arahan tersebut diselesaikan dalam masa terpendek, dan berapa lamakah untuk melaksanakan tugas jika diberi maklumat di bawah?

Loading time of an instruction (in ms) into different processors is as follows.

Masa untuk memuat satu arahan (dalam ms) ke dalam pemproses yang berbeza ialah seperti yang berikut.

Processor *Pemproses*	Time for loading instructions (in ms) *Masa untuk memuat arahan (dalam ms)*
A	2
B	1.5
C	1

Processing time (in ms) for each instruction in different processors is as follows.

Masa pemprosesan (dalam ms) bagi setiap arahan dalam pemproses yang berbeza ialah seperti yang berikut.

Processor *Pemproses*	Instruction v *Arahan v*	Instruction w *Arahan w*	Instruction x *Arahan x*	Instruction y *Arahan y*
A	10	6	8	10
B	8	9	6	7
C	12	5	7	9

Time (in ms) taken to transfer instructions between processors is as follows.

Masa (dalam ms) diambil untuk memindahkan arahan antara pemproses ialah seperti yang berikut.

To/*Dari* From/*Kepada*	A	B	C
A	0	2	1
B	1	0	1
C	2	2	0

(60/100)

(b) You want to put your valuable items into a safe box in a bank. The safe box can hold maximum 27kg. You have the items A, B, C and D in different quantities. See table below.

Anda mahu meletakkan barangan berharga dalam sebuah kotak keselamatan. Kotak keselamatan boleh menyimpan maksimum 27 kg. Anda mempunyai barang A, B, C dan D dalam kuantiti yang berbeza. Lihat jadual di bawah.

Item *Barang*	Quantity *Kuantiti*	Weight (kg/item) *Berat (kg/barang)*	Value/ item *Nilai/barang*
A	5	3 kg	$ 800
B	4	4 kg	$ 1000
C	3	5 kg	$ 1100
D	3	6 kg	$ 1400

(i) Describe an efficient approach using backtracking strategy to put the items into the safe box such that it holds the highest value.

Terangkan suatu pendekatan yang cekap dengan strategi pematahbalikan untuk memasukkan barang-barang berkenaan ke dalam kotak keselamatan supaya kotak berkenaan mengandungi nilai tertinggi.

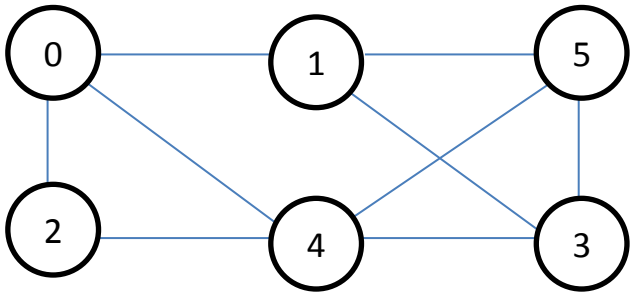
(ii) How much in value the safe box can hold maximally and what are the items in the safe box?

Apakah nilai maksimum kotak keselamatan berkenaan dapat menyimpan, dan apakah barang-barang dalam kotak keselamatan tersebut?

(40/100)

3. (a) Based on the graph below, by adding or deleting edge(s) of the original graph:

Berdasarkan graf di bawah, dengan menambah atau menghapuskan tepi(-tepi) graf asal:



(i) Make it a disconnected graph.

Jadikannya sebuah graf tak terkait.

(ii) Make it a complete graph.

Jadikannya sebuah graf lengkap.

(iii) Make it a bipartite graph.

Jadikannya sebuah graf dwipihak.

(iv) Make it a non-planar graph (Note: The graph is a planar graph).

Jadikannya sebuah graf bukan sesatah (Catatan: Graf berkenaan merupakan graf sesatah).

(v) Create a subgraph of the graph.

Cipta sebuah subgraf bagi graf berkenaan.

(25/100)

(b) Given below is the pseudo-code for finding a minimum spanning tree based on the Prim's algorithm.

Diberikan di bawah pseudokod untuk mencari sebuah pepohon rentang minimum berdasarkan algoritma Prim.

```
1  procedure Prim(G, w, Parent[0:n-1])
2    for v ← 0 to n-1 do
3         Nearest[v] ← ∞
4         InTheTree[v]← .false.
5    endfor
6    Parent[r]← -1
7    Nearest[r] ← 0
8    for Stage← 1 to n-1 do
9         Select vertex u that minimises Nearest[u] over all u such that
10             InTheTree[u] = .false.
11        InTheTree[u] = .true.
12        for each vertex v such that uv ϵ E do
13             if .not. InTheTree[v[  then
14                  if w(uv) < Nearest[v] then
15                       Nearest[v]← w(uv)
16                       Parent[r] ← u
17                  endif
18             endif
19        endfor
20   endfor
21   end Prim
```

(i) Why is this algorithm considered to be a greedy method?

Mengapakah algoritma ini dianggap sebagai kaedah tamak?

(ii) The algorithm terminates after only n-1 stages even though there are n vertices in the final minimum spanning tree. Why?

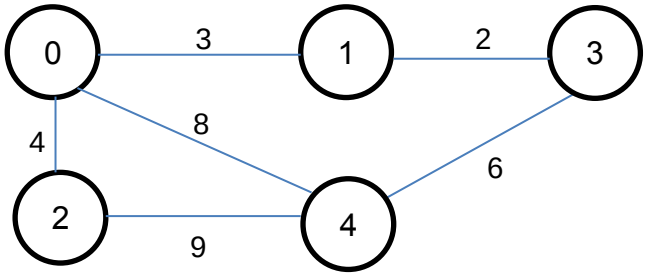
Algoritma berkenaan tamat selepas n-1 tahap walaupun terdapat n bucu dalam pepohon rentang minimum yang muktamad. Mengapa?

(iii) Based on the pseudo-code above, state the complexity of this algorithm and indicate which part of the pseudocode actually contributes to this complexity.

Berdasarkan pseudokod di atas, nyatakan kekompleksan algoritma ini dan tunjukkan bahagian manakah dalam pseudokod berkenaan yang sebenarnya menyumbang kepada kekompleksan ini.

(iv) Trace step by step how the algorithm works on the following graph.

Surih langkah demi langkah bagaimana algoritma berkenaan dilaksanakan ke atas graf berikut.



(45/100)

(c) Connected components of a graph basically partition the vertex set, i.e. a vertex u belongs to the strongly connected components containing v if and only if u belong to both $T_{out,v}$ and $T_{in,v}$ where $T_{out,v}$ is DFS (Depth First Search) out-tree and $T_{in,v}$ is DFS in-tree of the graph.

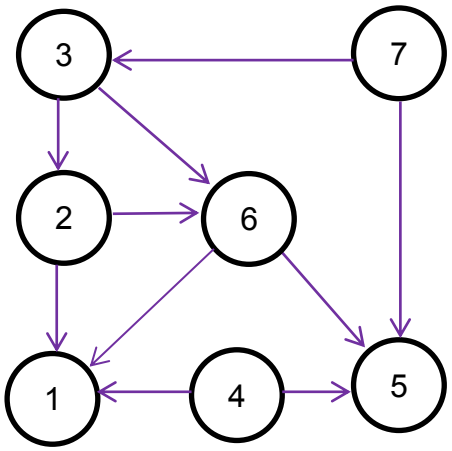
Komponen terkait sesebuah graf pada asasnya memetakkan set bucu, iaitu sebuah bucu tergolong dalam sebuah komponen terkait secara kuat yang mengandungi v jika dan hanya jika u berada dalam kedua-dua $T_{out,v}$ dan $T_{in,v}$, dan $T_{out,v}$ ialah pepohon keluar DFS (Gelintaran Kedalaman Dahulu) dan $T_{in,v}$ ialah pepohon ke dalam DFS graf berkenaan.

(i) Define $T_{out,v}$ and $T_{in,v}$.

Takrifkan $T_{out,v}$ *dan* $T_{in,v}$.

(ii) Obtain a strongly connected component of the following graph using the above mentioned method.

Dapatkan sebuah komponen terkait secara kuat graf berikut menggunakan kaedah yang tersebut di atas.



(30/100)

4. (a) Given the following pseudo-code for the Rabin Karp string matching algorithm:

Diberi pseudokod berikut untuk algoritma pemadanan rentetan Rabin Karp:

```
1  function KarpRabinStringMatcher(P[0:m-1],T[0,n-1],k,q)
2       c ← k^(m-1) mod q
3       Ṕ^(q) ← 0
4       Ť_s^(q) ← 0
5       for i ← 1 to m do
6            Ṕ^(q) ← (k*Ṕ^(q) + p[i]) mod q
7            Ť_s^(q) ← (k*Ť_s^(q) +T[i]) mod q
8       endfor
9       for s ← 0 to n - m do
10           if s > 0 then
11                Ť_s^(q) ← (k* Ť_s-1^(q) – T[s]*c) + T[s + m]) mod q
12           endif
13           if Ť_s^(q) = Ṕ^(q) then
14                if T_s = P then
15                     return(s)
16                endif
17           endif
18      endfor
19      return(0)
20 end KarpRabinStringMatcher
```

(i) Identify which part of the pseudocode applies Horner's rule in the computation and discuss its significance.

Kenal pastikan bahagian manakah dalam pseudokod berkenaan yang menerapkan peraturan Horner dan bincangkan kepentingannya.

(ii) Why does the **mod** (modulo) operator used in above algorithm?

Mengapakah pengendali **mod** *(modulo) digunakan dalam algoritma di atas?*

(iii) Line 13 of the pseudocode is the statement that checks whether a match has occurred. Why do we still need Line 14 to confirm it?

Baris 13 pseudokod berkenaan merupakan kenyataan yang menyemak sama ada suatu pemadanan telah berlaku. Mengapakah kita masih memerlukan Baris 14 untuk mengesahkannya?

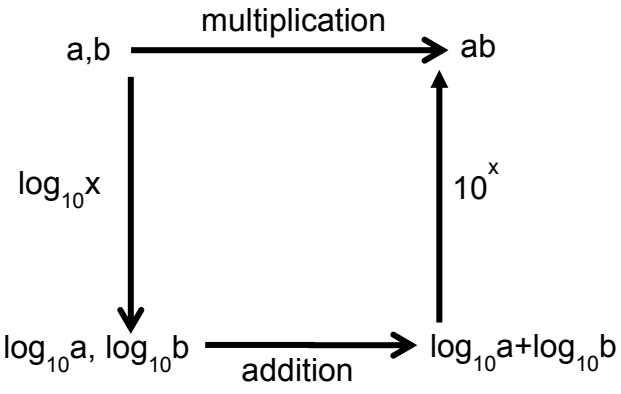
(iv) Show that worst-case complexity of the algorithm is O(mn).

Tunjukkan bahawa kekompleksan kes terburuk algoritma berkenaan ialah O(mn).

(50/100)

(b) Given below is the diagram that depicts the problem of transforming a problem into equivalent problem:

Diberi di bawah gambar rajah yang menggambarkan masalah menjelmakan suatu masalah ke dalam masalah setara:



(i) The diagram is said to commutative. Why?

Gambar rajah berkenaan dikatakan kalis tukar tertib. Mengapa?

(ii) In a more generic commutative diagram, Discrete Fourier Transform (DFT) can be used. Identify which parts of the above diagram respectively correspond to the problem in the original domain, the solution in the original domain, where DFT is used, and where inverse DFT is used.

Dalam gambar rajah yang lebih generik, Jelmaan Fourier Diskret (DFT) boleh digunakan. Kenal pastikan bahagian manakah dalam gambar rajah di atas masing-masing sepadan dengan masalah dalam domain asal, penyelesaian dalam domain asal, tempat DFT digunakan dan tempat DFT songsang digunakan.

(30/100)

(c) A generic nondeterministic polynomial (NP) algorithm is given as follows:

Sebuah algoritma polinomial tak berketentuan (NP) generik diberi seperti yang berikut:

```
function NPAlgorithm(A,I)
// A - a decision problem, I - an instance of problem A
1       1. In polynomial time, guess a candidate C for the problem A.
2       2. In polynomial time, use C to deterministically verify that I is a yes
3          instance.
4       if a yes instance is verified in step 2 then
5           return("yes")
6       else
7           return("don't know")
8       endif
9  end NPAlgorithm
```

(i) What is the role of the parameter A (a decision problem) in implying a polynomial solution to the original problem?

Apakah peranan parameter A (masalah keputusan) dalam mengimplikasikan penyelesaian polynomial kepada masalah yang asal?

(ii) Using your own words, what does **return**("yes") in line 5 imply?

*Dengan menggunakan perkataan anda sendiri, apakah yang diimplikasikan oleh **return**("yes") pada Baris 5?*

(20/100)