Second Semester Examination
2016/2017 Academic Session

June 2017

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 **NINE** printed pages before you begin the examination.

  *[Sila pastikan bahawa kertas peperiksaan ini mengandungi **EMPAT** soalan di dalam **SEMBILAN** 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. The **travelling salesperson problem** (**TSP**) asks the following question: "Given a list of n cities and the distances between each pair of cities, what is the shortest possible route that visits each city exactly once and returns to the origin city?"

Masalah perjalanan jurujual (MPJ) *bertanya soalan berikut: "Diberi suatu senarai n bandar dan jarak di antara setiap pasangan bandar-bandar, apakah laluan yang paling singkat yang akan melawat setiap bandar sekali sahaja dan kembali ke bandar asal?"*

(a) A naïve/straightforward algorithm can be implemented to find out all the combinations of routes in a TSP, and then the shortest route can be chosen.

Algoritma naif/mudah boleh dilaksanakan untuk mencari semua kombinasi laluan dalam MPJ, dan laluan paling dekat boleh dipilih.

(i) What are **two** (**2**) main primitive operations used in the algorithm.

*Apakah **dua** (**2**) operasi primitif yang digunakan dalam algoritma tersebut?*

(ii) What is the time complexity of the naïve algorithm? Describe how you get the answer.

Apakah kekompleksan masa algoritma naif berkenaan? Terangkan bagaimana anda mendapat jawapan tersebut.

(15/100)

(b) One possible approach to be applied on TSP is using the greedy algorithm.

Salah satu pendekatan yang mungkin digunakan pada MPJ ialah menggunakan algoritma tamak.

(i) What is greedy algorithm?

Apa itu algoritma tamak?

(ii) Describe a greedy algorithm that can give a reasonable good solution to TSP. Use a figure to help you to explain the algorithm.

Terangkan algoritma tamak yang boleh memberikan penyelesaian yang munasabah kepada MPJ. Gunakan gambar rajah bagi membantu anda menjelaskan algoritma tersebut.

(iii) What is the time complexity of the algorithm? Describe how you get the answer.

Apakah kekompleksan masa algoritma tersebut? Terangkan bagaimana anda mendapat jawapan tersebut.

(35/100)

(c) Another way to solve TSP is using convex hull algorithm.

Cara lain untuk menyelesaikan MPJ ialah menggunakan algoritma hul cembung.

(i) Describe the convex hull algorithm. Use a figure to help you to explain the algorithm.

Terangkan algoritma hul cembung. Gunakan gambar rajah untuk membantu anda untuk menerangkan algoritma tersebut.

(ii) What is the heuristic/assumption made when using convex hull algorithm to solve TSP?

Apakah heuristik/andaian yang dibuat apabila menggunakan algoritma hul cembung untuk menyelesaikan MPJ?

(iii) Explain the modification required on the convex hull algorithm to solve the TSP.

Terangkan pengubahsuaian yang diperlukan pada algoritma hul cembung untuk menyelesaikan MPJ.

(iv) What is the time complexity of the new algorithm? Describe how you get the answer.

Apakah kekompleksan masa algoritma hul cembung? Terangkan bagaimana anda mendapat jawapan tersebut.

(50/100)

2. The maximum subarray problem is the task of finding the contiguous subarray within a one-dimensional array of numbers which has the largest sum. For example, for the sequence of values −2, 1, −3, 4, −1, 2, 1, −5, 4; the contiguous subarray with the largest sum is 4, −1, 2, 1, with sum 6.

Masalah sub tatasusunan maksimum ialah masalah mencari sub tatasusuan berdampingan dalam suatu tatasusunan satu dimensi nombor yang mempunyai jumlah nilai terbesar. Sebagai contoh, untuk urutan nilai -2, 1, -3, 4, -1, 2, 1, -5, 4; sub tatasusunan berdampingan dengan jumlah terbesar ialah 4, -1, 2, 1, dengan jumlah 6.

(a) Could you suggest one possible area where the algorithm to solve the maximum subarray problem can be applied?

Bolehkah anda cadangkan satu bidang yang mungkin di mana algoritma untuk menyelesaikan masalah sub tatasusunan maksimum boleh digunakan?

(20/100)

(b) If we simply write a naïve/straight forward algorithm to find the maximum subarray.

Jika kita hanya menulis algoritma naif/mudah untuk mencari sub tatasusunan maksimum.

(i) What are the **two** (**2**) main primitive operations used in the algorithm?

*Apakah **dua** (**2**) operasi primitif utama yang digunakan dalam algoritma tersebut*?

(ii) Describe the naive algorithm.

Terangkan algoritma naif tersebut.

(iii) What is the complexity of the algorithm?

Apakah kekompleksan algoritma tersebut?

(40/100)

(c) A more efficient approach is to use dynamic programming to solve the problem.

Pendekatan yang lebih cekap adalah dengan menggunakan pengaturcaraan dinamik untuk menyelesaikan masalah tersebut.

(i) Explain how dynamic programming achieves the computational efficiency.

Terangkan bagaimana pengaturcaraan dinamik mencapai kecekapan pengiraan.

(ii) Describe an algorithm using dynamic programming to solve the maximum subarray problem. Use figure to explain the algorithm.

Terangkan suatu algoritma menggunakan pengaturcaraan dinamik untuk menyelesaikan masalah sub tatasusunan maksimum. Gunakan gambar rajah untuk menjelaskan algoritma tersebut.

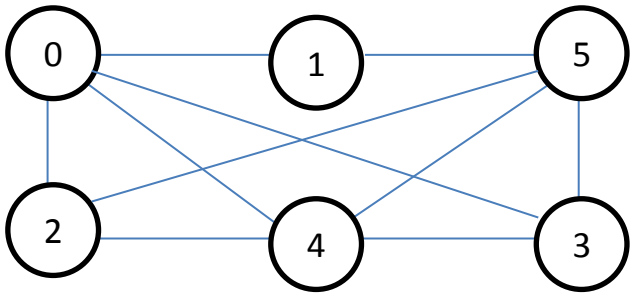
(iii) What is the complexity of the algorithm?

Apakah kekompleksan algoritma tersebut?

(40/100)

3. (a) Based on the graph below, by only deleting edge(s) (delete as minimum as possible):

Berdasarkan graf di bawah, dengan hanya menghapuskan tepi(-tepi) (hapuskan seminimum yang mungkin):



(i) Make it a disconnected graph.

Jadikannya sebuah graf tak terkait.

(ii) Make it a bipartite graph.

Jadikannya sebuah graf dwipihak.

(iii) Make it a planar graph.

Jadikannya sebuah graf sesatah.

(iv) Create a subgraph of the graph.

Cipta sebuah subgraf bagi graf berkenaan.

(v) Create a spanning subgraph of the graph.

Cipta sebuah subgraf rentang bagi graf berkenaan.

(25/100)

(b) Given below is the pseudocode for finding the minimum spanning tree based on the Kruskal's algorithm.

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

```
1   procedure Kruskal(G, w, MCSTree)
2   Sort the edges in non-decreasing order of weight w(e1) ≤ ..... ≤ w(em)
3   Forest ← Ø
4   Size ← 0
5   for i← 0 to n-1 do Parent[i] ← -1
6   endfor
7   j ←0
8   while Size ≤ n – 1 .and. j ≤ m do
9         j ← j+1
10        Find2(Parent[0:n-1], uj, r)  // ej =uj vj
11        Find2(Parent[0:n-1], vj, s)
12        if r ≠ s then    //add edge ej to Forest
13            Forest ← Forest U {ej}
14            Size ← Size +1
15            Union(Parent[0:n-1], r, s) // combine sets containing uj andvj
16      endif
17  endwhile
18  MCSTree← Forest
19  end Kruskal
```

(i) Is this algorithm considered to be a greedy method? Explain.

Adakah algoritma ini dianggap sebagai kaedah tamak? Jelaskan.

(ii) What is the role of Find2 in the above pseudocode (Lines 10 and 11)?

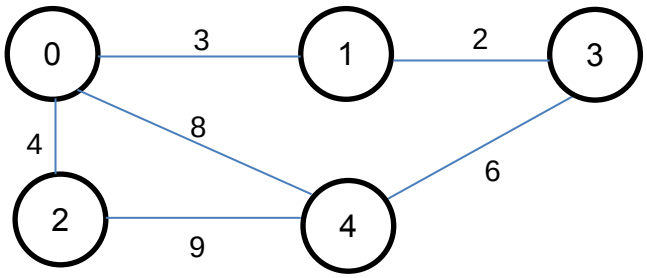
Apakah peranan Find2 dalam pseudokod di atas (Baris 10 dan 11)?

(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.



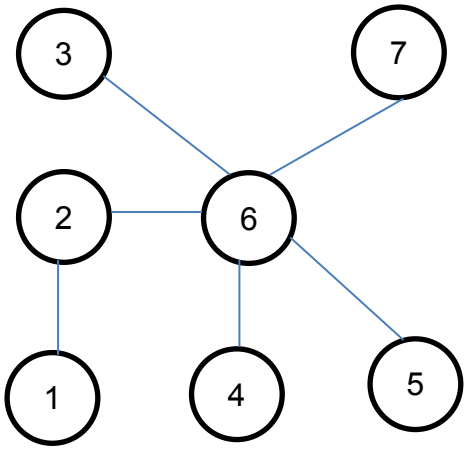
(45/100)

(c) (i) Define articulation point formally and explain its importance in a communication network.

Takrifkan titik artikulasi secara formal dan huraikan kepentingannya dalam rangkaian komunikasi.

(ii) Obtain articulation point(s) of the following graph.

Dapatkan titik(-titik) artikulasi bagi graf berikut.



(iii) Obtain bi-connected component(s) of the graph in Question 3(c)(ii) based on the articulation point(s) that you have obtained in that question.

Dapatkan komponen(-komponen) bagi graf dalam Soalan 2(c)(ii) berdasarkan titik(-titik) yang anda telah dapat dalam soalan berkenaan.

(30/100)

4. (a) Given the following pseudocode for the naive string-matching algorithm:

Diberi pseudokod berikut untuk algoritma pemadanan rentetan naif:

```
1   function NaiveStringMatcher(P[0:m-1],T[0:n-1])
2   P[0:m-1] // a pattern string of length m
3   T[0:n-1]  // a text string of length n
4   for s ← 0 to n-m do
5       if T[s : s + m – 1] = P then
6           return (s)
7       endif
8   endfor
9   return(-1)
10  end NaiveStringMatcher
```

(i) Explain the role of line 5 in the above pseudocode.

Huraikan peranan baris 3 dalam pseudokod di atas.

(ii) Obtain that best-case and worst-case complexities of the algorithm based on for loop (lines 4-8).

Dapatkan kekompleksan kes terbaik dan kes terburuk algoritma berkenaan berdasarkan gelung for (baris 4-18).

(iii) Give examples of P and T that will give the worst-case complexity.

Beri contoh P dan T yang akan memberikan kekompleksan kes terbaik.

(iv) Compare and contrast algorithmically and in terms of complexity, this algorithm with the Knuth-Morris-Pratt Algorithm string-matching algorithm.

Banding dan bezakan dari segi algoritma dan kekompleksan, algoritma ini dengan algoritma pemadanan rentetan Knuth-Morris-Pratt.

(50/100)

(b) Without going into mathematical detail or writing any code, explain how divide-and-conquer strategy is used in the Fast Fourier Transform and how this strategy yields O(n log n) complexity.

Tanpa perincian matematik atau menulis kod, huraikan bagaimana strategi bahagi-dan-tawan digunakan dalam Jelmaan Cepat Fourier dan bagaimana strategi ini menghasilkan kekompleksan O(n log n).

(25/100)

(c) (i) How is a digraph used in ranking of web pages in the page rank algorithm? Also, explain how the web pages are organised as a digraph.

Bagaimanakah graf berarah digunakan dalam pemeringkatan halaman web dalam algoritma taraf halaman? Juga, huraikan bagaimana halaman-halaman web berkenaan disusun sebagai sebuah graf berarah.

(ii) The PageRank, R[p] has an interesting interpretation in terms of random walk. B can be thought of as the matrix for a random walk on diagraph W, where B[p,q] is the probability that a random walker at page p ("aimless surfer") will follow the hyperlink from page p to page q. How does this concept of "aimless surfer" provide a meaningful interpretation of ranking of Web pages?

Peringkat Halaman, R[p] mempunyai penafsiran yang menarik dari segi perjalanan rawak. B boleh dianggap sebagai matriks untuk sebuah perjalanan rawak ke atas graf berarah W, dan B[p,q] adalah kebarangkalian seseorang pejalan rawak di halaman p ("pelayar tak bertujuan") akan melalui pautan hiper dari halaman p ke halaman q. Bagaimanakah konsep '"pelayar tak bertujuan' ini memberikan penafsiran yang bermakna dalam pemeringkatan laman web?

(25/100)