UNIVERSITI SAINS MALAYSIA

Supplementary Examinations
[Peperiksaan Kursus Semasa Cuti Panjang]

Academic Session 2007/2008
[Sidang Akademik 2007/2008]

June 2008
[Jun 2008]

CPT212/CPT201 – Design & Analysis of Algorithms
[Reka Bentuk & Analisis Algoritma]

Duration : 2 hours
[Masa : 2 jam]

INSTRUCTIONS TO CANDIDATE:
[ARAHAN KEPADA CALON:]

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

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

- Answer any **FOUR** questions only.

  *[Jawab mana-mana **EMPAT** soalan sahaja.]*

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

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

---

1. (a) (i) An algorithm is required to insert an element x into the first cell of an array A of size n and the array is not full. The algorithm is to shift the existing elements one position to the right in order to create a space for the new element in the first cell. Write a function in C++/Java that performs this task.

(ii) Determine the time complexity of your function in Question 1(a)(i) above for the best and worst cases in terms of n. Justify your answers.

(iii) What would be the time complexity of the algorithm in Question 1(a)(i) above if insertion is carried out at the end of the array i.e. the new element is now the last element of the array. Explain you answer.

(40/100)

(b) Given below is the selection-sort algorithm:

```
template<class T>
void selectionsort (data[], int n)
{     for (int i=0,j,least; i<n-1;i++){
            for (int j=i+1, least=i; j<n; j++)
                  if (data[j] < data[least])
                        least = j;
                  swap (data[least], data[i]);
      }
}
```

(i) Give the time complexity of the algorithm for best, average, and worst cases. Justify your answers.

(ii) Insert counter statements into the above code to count the number of swaps and the number of comparisons.

(30/100)

(c) Trace mergesort algorithm as it sorts the following array:

30 10 14 21 20

(30/100)

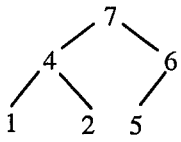
2. (a) Compare and contrast the worst case performances of binary trees and AVL trees in terms of searching, traversal, insertion, and deletion operations, and also Treesort algorithm.

(20/100)

(b) Why is it generally quite a problem in finding efficient implementation of a priority queue? Discuss your answer in terms of the operations, and the nature of the arrival and departure of the elements, and the priority criteria.

(35/100)

(c) Write a pseudocode (C++/Java) that will output the first element for each level of a heap beginning with the first level i.e. the root. For example the following heap and its array representation will be output as: 7 4 1



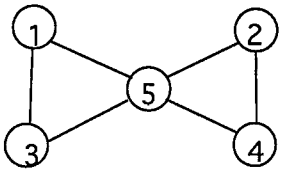
7	4	6	1	2	5	.....	

(30/100)

(d) Perform heapsort on the heap given in Question 2(c) above.

(15/100)

3. (a) Given the following graph:



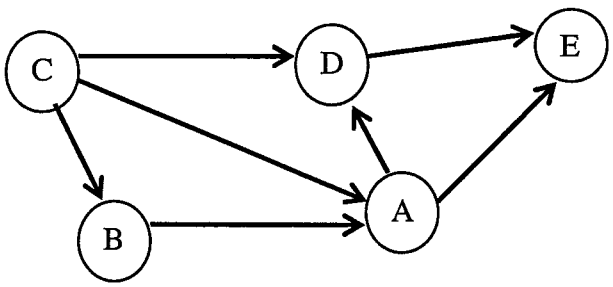
(i) Give the adjacency lists representation of the graph with the list of nodes represented as a linked list.

(ii) Give the adjacency lists representation of the graph with the list of nodes represented as an array.

(iii) What are the advantages and disadvantages of each of the variations of the adjacency lists representation as given in Questions 3(a)(i) and 3(a)(ii) above in terms of space and time complexities?

(35/100)

(b) Show the steps involved in carrying out topological sort on the graph below using the depth-first search approach (give only one of the possible outputs):



(25/100)

(c) A hash table uses a hash function *hash(key) = the middle letter of the key % size of table* and the keys are assumed to be of the same length and the length of the key is an odd number.

(i) Is the hash function a good hash function? Justify your answer. Give also a pattern that hashes into the same location for the hash function.

(ii) Why is it always recommended that *size of table* in the above hash function should be a prime number?

(iii) Write a function in C++/Java that implements the above hash function.

(40/100)

4. (a) Why does a B-tree operate closely with secondary storage and how can it be tuned to reduce the impediments imposed by such storage?

(35/100)

(b) Given below is a code for an algorithm for finding a key in a B tree:

```
BTreeNode *BTreeSearch(keyType K, BTreeNode *node) {
     if (node != 0) {
           for (i=1; i <= node->KeyTally &&
                 node.keys[i-1] < K; i++);
           if (i>node->keyTally || node->keys[i-1] >K)
                 return BTreeSearch(K,node->pointers[i-1]);
           else return node;
     }else return 0;
}
```

(i) What is the purpose of the "for" loop in the above code?

(ii) What is the purpose of the second "if" statement in the above code?

(iii) Modify the above code so that the search starts from the last key of each node instead of from the first key of the node.

(45/100)

(c) Compare and contrast the 2-4 tree and the binary tree in terms of shape, flexibility and efficiency.

(20/100)

5. (a) Name the sequential-fit method that you would prefer for each of the following situations:

(i) The free blocks are arranged by size in ascending order.

(ii) The free blocks are arranged by size in descending order.

(iii) The free blocks are ordered by address.

Justify your answer.

(30/100)

(b) You are given five letters A, B, C, D, and E with probabilities 0.05, 0.25, 0.4, 0.2, and 0.1 respectively.

(i) Create a Huffman tree for the above letters.

(ii) How many different Huffman trees of the same average length for the above letters can be generated? Explain your answer.

(30/100)

(c) Given below a pseudocode for a simple approach to string matching:

```
bruteForceStringMatching(pattern P, text T)
    i = 0;
    while i ≤ |T| - |P|
        j = 0;
        while Ti == Pj and j < |P|
            i++;
            j++;
        if j == |P|
            return match at i - |P|;
        i = i - j + 1;
    return no match;
```

(i) Why is the above algorithm considered to be a straightforward algorithm? Discuss your answer by referring to the above code.

(ii) Where in the above algorithm should we insert the counter statements to count the number of comparisons if we are interested in the number of comparisons performed by the algorithm?

(40/100)

1. (a) (i) Sebuah algoritma diperlukan untuk menyisip sebuah unsur x ke dalam sel pertama sebuah tatasusunan A bersaiz n dan tatasusunan tersebut tidak penuh. Algoritma berkenaan perlu menganjak semua unsur sedia ada satu kedudukan ke kanan untuk mencipta satu ruang untuk unsur baru berkenaan di sel yang pertama. Tulis fungsi dalam C++/Java yang melakukan tugas tersebut.

(ii) Tentukan kekompleksan masa untuk fungsi anda dalam Soalan 1(a)(i) di atas bagi kes terbaik dan terburuk dalam sebutan n. Justifikasikan jawapan anda.

(iii) Apakah kekompleksan algoritma dalam Soalan 1(a)(i) di atas jika penyisipan dilakukan pada hujung tatasusunan iaitu unsur baru berkenaan sekarang menjadi unsur terakhir tatasusunan.

(40/100)

(b) Diberi di bawah algoritma isihan pilih:

```
template<class T>
void selectionsort (data[], int n)
{     for (int i=0,j,least; i<n-1;i++){
            for (int j=i+1, least=i; j<n; j++)
                  if (data[j] < data[least])
                        least = j;
                  swap (data[least], data[i]);
      }
}
```

(i) Beri kekompleksan masa algoritma berkenaan bagi kes terbaik, purata, dan terburuk. Jelaskan jawapan anda.

(ii) Sisipkan kenyataan-kenyataan pembilang ke dalam kod di atas untuk menghitung bilangan penyilihan dan bilangan perbandingan.

(30/100)

(c) Surih algoritma isihan cantum apabila algoritma ini menyisih tatasusunan berikut:

30 10 14 21 20

(30/100)

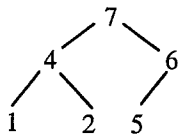
2. (a) Banding dan bezakan prestasi pepohon perduaan dan pepohon AVL dari segi pengendalian penggelintaran, penyusuran, penyisipan, dan penghapusan, dan algoritma pepohon.

(20/100)

(b) Mengapakah biasanya terdapat masalah dalam mencari pelaksanaan yang cekap bagi baris gilir keutamaan? Bincang jawapan anda dari segi pengendalian, dan bentuk ketibaan dan perlepasan unsur-unsur, dan kriteria keutamaan.

(35/100)

(c) Tulis pseudokod (C++/Java) yang memaparkan unsur pertama bagi setiap paras sebuah timbunan bermula dengan paras yang pertama iaitu akar. Sebagai contoh timbunan berikut dan perwakilan tatasusunannya akan dioutput sebagai: 7 4 1



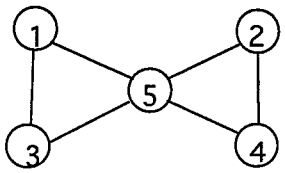
7	4	6	1	2	5	.....	

(30/100)

(d) Lakukan isihan timbun bagi timbunan yang diberikan dalam Soalan 2(c) di atas.

(15/100)

3. (a) Diberikan graf berikut:



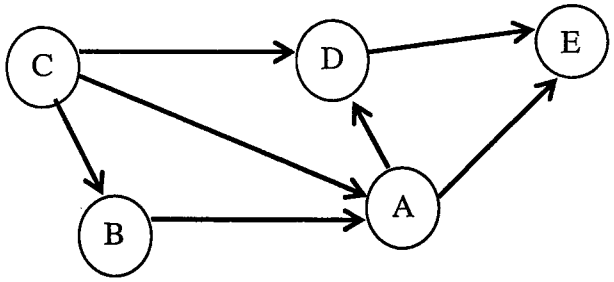
(i) Beri perwakilan senarai kesebelahan graf berkenaan dengan senarai nod diwakilkan sebagai senarai berpaut.

(ii) Beri perwakilan senarai kesebelahan graf berkenaan dengan senarai nod diwakilkan sebagai tatasusunan.

(iii) Apakah kebaikan dan keburukan setiap variasi perwakilan senarai kesebelahan yang diberikan dalam soalan 3(a)(i) dan 3(a)(ii) dari segi kekompleksan ruang dan masa?

(35/100)

(b) Tunjukkan langkah-langkah yang terlibat dalam menjalankan isihan topologi ke atas graf di bawah menggunakan pendekatan gelintaran kedalaman dahulu (beri hanya satu output yang mungkin):



(25/100)

(c) Sebuah jadual cincangan menggunakan fungsi cincangan *hash(key) = huruf tengah key % saiz jadual* dan kunci-kunci berkenaan dianggap mempunyai panjang yang sama dan panjang kunci adalah satu nombor ganjil.

(i) Adakah fungsi cincangan berkenaan baik? Jelaskan jawapan anda. Beri juga pola yang mencincang ke dalam lokasi yang sama bagi fungsi berkenaan.

(ii) Mengapakah biasanya disarankan supaya *saiz jadual* dalam fungsi cincangan di atas sepatutnya nombor perdana?

(iii) Tulis fungsi dalam C++/Java yang melaksanakan fungsi cincangan di atas.

(40/100)

4. (a) Mengapakah pepohon B beroperasi dengan rapat dengan storan sekunder dan bagaimanakah pepohon ini boleh disesuaikan untuk mengurangkan rintangan yang ada pada storan sedemikian?

(35/100)

(b) Diberi kod bagi algoritma untuk mencari suatu kunci dalam pepohon B:

```
BTreeNode *BTreeSearch(keyType K, BTreeNode *node) {
     if (node != 0) {
           for (i=1; i <= node->KeyTally &&
                 node.keys[i-1] < K; i++);
           if (i>node->keyTally || node->keys[i-1] >K)
                 return BTreeSearch(K,node->pointers[i-1]);
           else return node;
     }else return 0;
}
```

(i) Apakah tujuan gelung "for" dalam kod di atas?

(ii) Apakah tujuan kenyataan "if" yang kedua dalam kod di atas?

(iii) Ubah suai kod di atas supaya gelintaran bermula dari kunci terakhir bagi setiap nod dan bukannya dari kunci pertama nod.

(45/100)

(c) Banding dan bezakan pepohon 2-4 dengan peohon perduaan dari segi rupa bentuk, keluwesan dan kecekapan.

(20/100)

5. (a) Namakan kaedah penyuaian berjujukan yang akan anda lebih suka bagi setiap situasi berikut:

(i) Blok-blok bebas disusun mengikut saiz dalam tertib menaik.

(ii) Blok-blok bebas disusun mengikut saiz dalam tertib menurun.

(iii) Blok-blok bebas bertertib mengikut alamat.

Jelaskan jawapan anda.

(30/100)

(b) Anda diberikan lima huruf A, B, C, D, dan E dengan masing-masing kebarangkalian 0.05, 0.25, 0.4, 0.2, dan 0.1.

(i) Cipta satu pepohon Huffman untuk huruf-huruf di atas.

(ii) Berapa banyakkah pepohon Huffman yang berbeza yang sama panjang boleh dijana bagi huruf-huruf di atas? Huraikan jawapan anda.

(30/100)

(c) Diberikan di bawah pseudokod untuk pendekatan mudah pemadanan rentetan:

```
bruteForceStringMatching(pattern P, text T)
    i = 0;
    while i ≤ |T| - |P|
        j = 0;
        while Ti == Pj and j < |P|
            i++;
            j++;
        if j == |P|
            return match at i - |P|;
        i = i - j + 1;
    return no match;
```

(i) Mengapakah algoritma di atas dikatakan sebagai algoritma yang mudah? Bincangkan jawapan anda dengan merujuk kepada kod di atas.

(ii) Di manakah dalam algoritma di atas kita sepatutnya menyisipkan kenyataan-kenyataan pembilang untuk menghitung bilangan perbandingan jika kita berminat tentang bilangan perbandingan yang dilakukan oleh algoritma berkenaan?

(40/100)