UNIVERSITI SAINS MALAYSIA

Peperiksaan Semester Pertama
Sidang Akademik 1999/2000

September 1999

CTP201 - Reka Bentuk & Analisis Algoritma
CSC201 - Struktur Data & Algoritma

Masa : [3 jam]

ARAHAN KEPADA CALON:

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

- Jawab mana-mana **EMPAT** soalan dan semua jawapan hendaklah ditulis dalam Bahasa Malaysia.

...2/-

1. *Matriks jarang* adalah matriks yang mengandungi banyak unsur sifar di dalamnya. Satu matriks jarang yang besar dengan M unsur bukan sifar boleh disimpan di dalam 3 tatasusunan linear Baris, Lajur, dan Nilai dengan setiap tatasusunan mempunyai julat di antara 0 hingga M-1 (termasuk 0 dan M-1). Baris[k], Lajur[k] dan Nilai[k] adalah masing-masing baris, lajur dan nilai unsur bukan sifar yang ke-k mengikut susunan indeks *menaik* (mengikut baris dan kemudiannya lajur).

Contoh: Matriks Jarang -> Baris Lajur Nilai

$$\begin{bmatrix} 1 & 3 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} 1 \\ 1 \\ 3 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} \begin{bmatrix} 1 \\ 3 \\ 1 \end{bmatrix}$$

(a) Takrifkan sebuah *kelas* `SparseMatrix` dalam C++ untuk pelaksanaan matriks jarang yang terdiri daripada integer-integer seperti yang dihuraikan di atas. Sediakan fungsi *pembina* dan *pemusnah* yang membolehkan objek kelas ini diawalkan apabila diisytiharkan dan dimusnahkan apabila tidak diperlukan. Tatasusunan-tatasusunan diperuntukkan secara *dinamik*. Sediakan juga beberapa *fungsi anggota* (tanpa pelaksanaan) termasuklah fungsi-fungsi yang diberikan dalam (b) di bawah.

[20/100]

(b) Tulis *pelaksanaan* untuk fungsi-fungsi berikut bagi kelas di atas seperti yang dihuraikan di bawah. Beri juga *kekompleksan algoritma* bagi setiap fungsi tersebut

(i) print - *mencetak* matriks jarang dalam bentuk (Baris, Lajur, Nilai). Umpamanya bagi matriks contoh di atas, print akan mencetak :

(1,1,1)
(1,2,3)
(3,3,1)

(ii) trace - memberikan *hasil tambah* unsur-unsur *pepenjuru*. Umpamanya bagi matriks contoh di atas fungsi ini akan mengembalikan nilai 2.

(iii) element - mengembalikan *unsur [i][j]* matriks jarang berkenaan. Umpamanya bagi matriks contoh di atas element(1,3) mengembalikan nilai 0 dan element(1,2) mengembalikan nilai 3.

[40/100]

(c) Bincang kemungkinan untuk menukarkan kelas di atas sebagai *templat*. Jika mungkin beri takrifan templat tersebut.

[10/100]

(d) Tanpa menulis sebarang kod, bincang bagaimana *lebihan beban pengendali* boleh digunakan dalam kelas di atas.

[10/100]

(e) Tulis satu *pembina salinan* untuk kelas di atas dan beri kekompleksan algoritma bagi fungsi ini.

[20/100]

...3/-

2. (a) Diberikan algoritma *isihan gelembung* (seperti yang diberikan dalam kuliah) berikut:

```
void bubble(int a[], int N)
{
  int i, j;
  for (i = N; i >= 1; i--)
    for (j = 2; j <= i; j++)
      if (a[j-1] > a[j]) swap(a, j-1, j);
}
```

(i) Ubahsuai fungsi di atas seperti berikut:

Guna penggelembungan dua arah dalam setiap laluan. Dalam penggelembungan yang pertama, unsur terkecil digelembungkan *ke kiri*. Dalam penggelembungan yang kedua, unsur terbesar digelembungkan *ke kanan*. Pengisihan ini digelar isihan penggoncang (shaker sort).

(ii) Tentukan *kekompleksan* algoritma yang diubahsuaikan dalam (i) di atas dan tentukan juga sama ada pengubahsuaian algoritma berkenaan *memberi kesan* terhadap prestasi kes terburuk isihan gelembung.

[30/100]

(b) (i) Yang manakah antara jujukan berikut *timbunan*? Tunjukkan bagaimana anda mendapatkan jawapan anda.

42 35 37 20 14 18 7 10

42 35 18 20 14 30

20 20 20 20 20 20

(ii) Lukis sebuah pepohon yang juga sebuah *timbunan* dan pada masa yang sama sebuah *pepohon gelintaran perduaan*.

[25/100]

(c) (i) Beri huraian yang lengkap dalam perkataan anda sendiri tentang algoritma *isihan cantum* (Anda tidak perlu memberikan contoh dan kod atur cara).

(ii) Tunjuk langkah demi langkah bagaimana *isihan cantum* melakukan pengisihan ke atas set data berikut:

M E R G E S O R T I N G

(iii) Tunjuk langkah demi langkah bagaimana *cantuman berbilang hala terimbang* yang menggunakan cantuman *dua hala empat pita* melakukan pengisihan ke atas set data dalam (ii) di atas. Anggapkan ingatan komputer yang ada hanya boleh menampung *tiga* rekod sahaja.

(iv) Kaedah pengisihan luaran *cantuman berbilang hala terimbang* menggunakan konsep yang terdapat dalam kaedah pengisihan dalaman *isihan cantum*. Banding dan bezakan kedua-dua kaedah tersebut.

[45/100]

...4/-

3. (a) Diberikan algoritma *Gelintaran Kedalaman Dahulu* (senarai kesebelahan) (seperti yang diberikan dalam kuliah) berikut:

```
void search()
{
  int k;
  for (k = 1; k <= V; k++) val[k] = unseen;
  for (k = 1; k <= V; k++)
    if (val[k] == unseen) visit(k);
}
void visit(int k)
{
  struct node *t; val[k] = ++id;
  for (t = adj[k]; t != z; t = t->next)
    if (val[t->v] == unseen) visit(t->v);
}
```

Bincang bagaimana algoritma bagi setiap kes berikut dapat dilaksanakan dengan melakukan *pengubahsuaian* fungsi(-fungsi) di atas. Nyatakan dengan penjelasan *kekompleksan* dalam *tatatanda O Besar* bagi setiap kes.

(i) Menguji sama ada sesuatu graf *terkait.*

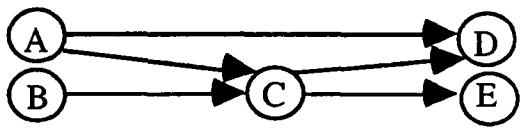
(ii) *Menghitung* bilangan *komponen terkait* sesebuah graf.

(iii) *Mencetak komponen-komponen terkait* sesebuah graf.

(iv) Melaksanakan *isihan topologi terbalik.*

[40/100]

(b) (i) Beri satu susunan *topologi terbalik* untuk graf berikut berdasarkan pengubahsuaian yang dibuat dalam (a)(iv) di atas.



(ii) Beri satu *contoh penggunaan praktik* isihan topologi terbalik.

[15/100]

(c) Anda telah mempelajari tiga perwakilan graf (matriks kesebelahan, senarai kesebelahan, senarai multipaut). Perwakilan manakah yang *paling mudah* (bukan paling pantas) untuk melaksanakan setiap algoritma berikut? Justifikasikan jawapan anda.

(i) Menentukan sama ada dua bucu *bersebelahan.*

(ii) Menentukan sama ada sesuatu bucu adalah *titik hujung* bagi sesuatu tepi.

(iii) Menghitung *bilangan tepi* di dalam sesebuah graf.

(iv) Menentukan semua tepi yang *berakhir* di sesebuah bucu.

(v) Algoritma *Warshall.*

(vi) Algoritma *Floyd.*

[45/100]

...5/-

4. (a) Pertimbangkan masalah *penggelintaran berjujukan* sesuatu kunci k di dalam tatasusunan n kunci.

(i) Adakah anda perlu membuat n *perbandingan* jika senarai *terisih* dan k *tidak wujud*? Jelaskan.

(ii) Apakah *kekompleksan* algoritma jika senarai *terisih* dan k *wujud*? Jelaskan dengan memberikan kes *terburuk*, kes *purata* dan kes *terbaik*.

(iii) Apakah *kekompleksan* algoritma jika senarai *tidak terisih* dan k *wujud*? Jelaskan dengan memberikan kes *terburuk*, kes *purata* dan kes *terbaik*.

(iv) Tulis satu *templat fungsi* dalam C++ untuk kes senarai terisih bagi masalah di atas.

[40/100]

(b) Nombor-nombor (integer) yang diberikan di bawah disimpan di dalam sebuah jadual dengan 7 lokasi. Fungsi cincangan menggunakan lokasi = nombor % 7, % adalah pengendali modulus. Nombor-nombor tersebut adalah 13, 5, 22, 8, 34, 19, 21 dan asalnya jadual cincang kosong.

(i) Guna *teknik rantaian berasingan* untuk menyelesaikan perlanggaran dengan membina jadual cincangan bagi nombor-nombor di atas.

(ii) Bincang dengan terperinci kebaikan dan keburukan penggunaan *pepohon gelintaran perduaan* sebagai ganti kepada senarai (rantai) berpaut dalam teknik rantaian berasingan.

(iii) Bagi nombor-nombor di atas, bina jadual cincang berasaskan huraian dalam (ii) di atas.

[30/100]

(c) (i) Huraikan konsep *pepohon AVL*.

(ii) Bagimanakah pepohon AVL *mengatasi* kelemahan pepohon gelintaran perduaan?

(iii) Apakah hasil penyisipan 1,2,3,....,8 ke dalam *pepohon AVL* yang asalnya kosong?

(iv) Secara umum, apakah yang berlaku ketika anda menyisipkan 1,2 3, ... ke dalam pepohon AVL atau dengan perkataan lain apakah nilai-nilai akar ketika proses penyisipan dilakukan berdasarkan jawapan anda dalam (iii) di atas?

[30/100]

...6/-

5. (a) Diberikan algoritma berikut yang mencari *kejadian pertama* satu corak p dalam rentetan teks a.

```
int brutesearch(char *p, char *a)
  {
    int i, j, M = strlen(p), N = strlen(a);
    for (i = 0, j = 0; j < M && i < N; i++, j++)
      if (a[i] != p[j]) { i -= j-1; j = -1; }
    if (j == M) return i-M; else return i;
  }
```

(i) Apakah kekompleksan *kes terburuk* bagi fungsi di atas? Jelaskan berserta dengan contoh.

(ii) Ubahsuaikan fungsi di atas supaya algoritma ini mengembalikan kedudukan mula kejadian pertama corak p di dalam rentetan teks a *pada* atau *selepas kedudukan k.*

(iii) Adakah *kekompleksan* fungsi di atas berubah setelah diubahsuai dalam (ii) di atas? Jelaskan.

[35/100]

(b) (i) *Pepohon huraian* merupakan satu struktur yang penting bagi pengkompil. Nyatakan peranan dan kegunaan pepohon huraian dalam proses pengkompilan.

(ii) Apakah fasa-fasa utama pengkompil yang memerlukan *jadual simbol*? Nyatakan *tiga* fasa utama sahaja dan huraikan peranan jadual simbol dalam setiap fasa yang anda diberikan.

[25/100]

(c) (i) Salah satu kelemahan teknik *penyuaian pertama* ialah penghasilan banyak blok kecil pada awal senarai bebas. Kenapakah ini terjadi? Bagaimanakah anda boleh mengatasi masalah ini?

(ii) Huraikan teknik *penyuaian terburuk* dan *ubahsuaikan* algoritma penyuaian terbaik di bawah kepada algoritma penyuaian terburuk.

```
p = freeblock;
rq = r = q = NULL;
rsize = memsize+1;
alloc = NULL;
while (p != NULL) {
      if (size(p) >= n && size(p) < rsize) {
            r = p; rq = q; rsize = size(p);
      }
      q = p; p = next(p);
}
if ( r!= null){
      alloc = r + rsize - n;
      if ( rsize == n )
            if (rq == null) freeblock = next(r);
            else next(rq) = next(r);
      else size(r) = rsize -n;
}
```

[40/100]

- ooo0ooo -