國立中央大學 107 學年度碩士班考試入學試題

所別： 資工類　　　　　　　　　　　　　　　　　　共8頁　第1頁

科目： 資料結構與演算法

本科考試禁用計算器　　　　　　　　　　　　　　*請在答案卷(卡)內作答

單選題 (每題 4 分，答錯倒扣一分)

1. Consider the following recursive sorting algorithm to sort an array A[ ] of n numbers:

```
E-SORT(A, i, j)
1.  if A[i] > A[j]
2.      then exchange A[i] ↔ A[j];
3.  if i + 1 ≥ j
4.      then return;
5.  k ← ⌊(j - i + 1)/3⌋;      // Round down.
6.  E-SORT(A, i, j - k);   // Sort first two-thirds.
7.  E-SORT(A, i + k, j);   // Sort last two-thirds.
8.  E-SORT(A, i, j - k);   // Sort first two-thirds again.
```

What is the time complexity of the algorithm if it is called as: E-SORT(A, 1, n)?
(a) $\Theta(n)$ (b) $\Theta(n\log n)$ (c) $\Theta(n^{\log_2 3})$ (d) $\Theta(n^2)$ (e) $\Theta(n^{\log_{3/2} 3})$

2. Follow the previous question. What is the amount of auxiliary space (extra space or temporary space not including the input array) used by the algorithm E-SORT?
(a) $\Theta(1)$ (b) $\Theta(\log n)$ (c) $\Theta(n^{1/3})$ (d) $\Theta(n^{2/3})$ (e) $\Theta(n)$

3. The following statements S_A and S_B about minimum spanning tree (MST) may not be correct. Assume that the weighted graph $G = (V, E)$ is undirected and connected. Do not assume that edge weights are distinct unless this is specifically stated. Consider the following two statements:
S_A: If the lightest edge in G is unique, then it must be part of every MST of G.
S_B: If G has a cycle with a unique lightest edge e, then e must be part of every MST of G.
(a) Both S_A and S_B are false. (b) Both S_A and S_B are true. (c) S_A is true and S_B is false. (d) S_A is false and S_B is true.

4. The Floyd-Warshall algorithm is a famous algorithm to solve the all-pairs shortest-paths problem on a weighted directed graph $G = (V, E)$. The algorithm gives a recursive formula to compute $d^{(k)}[i, j]$, the length of a shortest path from i to j using only vertices with indices $\leq k$. Here it is assumed that the vertex set $V = \{1, 2, \ldots, n\}$ if the given graph contains n vertices. Now if the given graph has 5 vertices and the edges: (1,2,3), (1,3,8), (1,5,−4), (2,4,1), (2,5,7), (3,2 4), (4,1,2), (4,3,−5), (5,4,6), where each triple (i, j, t) represents there is an edge directed

注意:背面有試題

參考用

from i to j with weight t. Then, after the execution of the algorithm, the value $d^{(4)}[2, 3]$ = (a) −4 (b) −2 (c) −1 (d) 2 (e) 3..

5. Follow the previous question. The value $d^{(5)}[1, 4]$ = (a) −4 (b) −2 (c) −1 (d) 2 (e) 3.

6. Given two sequences ABACBACBBAA and BDABCBACBDA, what is the length of the LCS (Longest Common Subsequence) of these two sequences? (a) 5 (b) 6 (c) 7 (d) 8 (e) 9.

7. Consider a variation of the towers of Hanoi problem. We no longer assume that all the disks are initially on one peg. They may be arbitrarily distributed among the three pegs, say pegs A, B, and C, as long as they are ordered in decreasing sizes on each peg. The purpose of the puzzle remains to move all disks to one specified peg, under the same constraints as the original problem, with as few moves as possible. We use array d[ ] to store such a distribution. For example, if d[1] = A, d[2] = C, d[3] = C, it means disc 1 (the disc of smallest size) is on peg A, and discs 2 and 3 are on peg C. If peg C is the destination peg, then only 1 move is needed. If peg A is the destination peg, then it needs 6 moves. Now, given d[1] = A, d[2] = C, d[3] = C, d[4] = B, d[5] = B, d[6] = A, how many moves are needed to move all 6 disks to peg C. (a) 35 (b) 36 (c) 37 (d) 38 (e) 39.

Suppose that array A[1:n] maintains a binary tree. For a binary tree node stored in A[i], its two children (if exist) are stored in A[2i] and A[2i+1], respectively. the program below aims to complete the following two tasks

(1) adjust array A to establish a max heap, and
(2) apply heap sort on the max heap built in (1) in nondecreasing order.

```
void adjust (int A[], int root, int n)
{
   int child, rootkey;
   int temp;
   temp = A[root];
   rootkey = A[root];
   child = 2* root;
   while ( child <=n ) {
      if (( child <=n ) && ( A[child] < A[child+1] ))
            child ++;
```

注意:背面有試題

參考用

國立中央大學107學年度碩士班考試入學試題

```
            if ( rootkey > A[child] )
                break;
            else {
                    ____(B1)______;
                  child *=2; }
        }
        ____(B2)______;
    }

    void heapsort (int A[], int n)
    { /* perform a heap sort on A[1:n] */
        int i, j;
        int temp;
        for ( __(B3)__; i>0; i++)       /* adjust the binary tree to establish the max heap */
            adjust (A, i, n);
        for (1=n-1; i>0; i--) {          /* heap sort */
            swap( A[1], A[i+1], temp);     /* exchange A[i] and A[i+1] */
            adjust(A, 1, i); }
    }
```

8. Blank (B1) in the program above should be (a) A[child] = A[root] (b) A[child/2] = A[root] (c) A[child/2] = A[child] (d) A[root] = A[root*2] (e) A[root] = A[child]

9. Blank (B2) in the program above should be (a) A[child] = temp (b) A[root] = A[child] (c) A[child] = A[child/2] (d) A[child/2] = temp (e) A[root] = temp

10. Blank (B3) in the program above should be (a) i=1 (b) i=0 (c) i=n/2 (d) i=n (e) i=n/2+1

單選題 (每題5分，答錯倒扣一分)

Program quickSort below aim to apply quick sort to A[left: right] into non-decreasing order.

```
    void quickSort ( int A[], int left, int right )
    {
        int pivot, i, j;
```

注意:背面有試題

參考用

```
        int temp;
        if ( left < right ) {
            i = left-1; j = right;
            pivot = A[right];
            do {
                    do i++; while ( A[i] < pivot );
                    do j--; while ( A[j] > pivot );
                    If (____(B4)____) swap(A[i], A[j], temp);    /* exchange A[i] and A[j] */
            } while (____(B5)____);
            swap (____(B6)_____);
            quickSort( A, left, j-1 );
            quickSort( A, j+1, right ); }
    }
```

11. Blank (B4) in the algorithm above should be (a) i>right-1 (b) j<left+1 (c) i>j (d) i>=j (e) i<j

12. Blank (B5) in the algorithm above should be (a) i>right-1 (b) j<left+1 (c) i>j (d) i>=j (e) i<j

13. Blank (B6) in the algorithm above should be (a) A[left], A[j], temp (b) A[i], A[right], temp (c) A[right], A[j], temp (d) A[left+1], A[right], temp (e) A[right-1], A[j], temp

14. A hash function maps a key into a bucket in the hash table. Which of the following statements is false? (a) A Biased use of the hash table is not a desired property of hash functions. (b) A division hash function with divisor $D = 2^r$, where r is an integer, may result in serious collision. (c) Doubling hash table (i.e., rebuild the hash table by doubling the number of table buckets) may result in key search suspended for unacceptable long periods while the rebuild is in progress. (d) Dynamic hash reduces the table rebuild time by ensuring a small amount of rebuild changes (e) When chaining is used to resolve overflows, the search for a key involves comparison with keys that have different hash values.

注意:背面有試題

參考用

國立中央大學107學年度碩士班考試入學試題

所別： 資工類

科目： 資料結構與演算法

本科考試禁用計算器 ＊請在答案卷(卡)內作答

多選題(每題5分，答錯每小題倒扣1分，扣到零分為止)

15. Below is the Maximum Contiguous Subsequence Sum Dynamic Programming (MCSSDP) algorithm returning the all-element sum of a **non-empty** contiguous subsequence S' of a given sequence S such that the sum is maximum, where $S=\{s_1,\ldots,s_n\}$, $n>0$, is a sequence of positive or negative integers. Choose all items that all together allow S' to be an **empty** subsequencc, whose sum is assumed to be 0.

(a) delete both line 1 and line 2

(b) replace line 3 with $m \leftarrow \max(s_1, 0)$

(c) replace line 6 with $m \leftarrow \max(m+s_i, 0)$

(d) replace line 7 with $mcss \leftarrow \max(m+s_i, 0)$

(e) replace line 7 with $mcss \leftarrow \max(m, s_i, 0)$

Algorithm MCSSDP

Input: a non-empty sequence $S=\{s_1,\ldots,s_n\}$, $n>0$, where s_i , $1\le i\le n$, is a positive or negative number

Output: the maximum all-element sum of a non-empty contiguous subsequence of S

```
1. if all elements of $S$ is negative then
2.     return max($s_1,\ldots, s_n$)
3. $m \leftarrow s_1$
4. $mcss \leftarrow m$
5. for $i \leftarrow 2$ to $n$ do
6.     $m \leftarrow \max(m+s_i, s_i)$
7.     $mcss \leftarrow \max(mcss, m)$
8. return $mcss$
```

參考用

16. Choose the correct statement(s):

(a) If we can solve a prolem X with a polynomial time-complexity non-deterministic algorithm, then X is an NP problem.

(b) If X is an NP-hard problem, then every NP problem can polynomially reduce to X.

(c) Let X be an NP-hard problem. If X can be solved by a deterministic algorithm with polynomial time complexity in the worst case, then every NP probloem can be solved by a deterministic algorithm with polynomial time complexity in the worst case.

(d) Let X be an NP-hard problem. If a problem Y can polynomially reduce to X, then Y is

注意:背面有試題

NP-hard.

(e) Let X be an NP-hard problem. If X can polynomially reduce to a problem Y, then Y is NP-hard.

17. Choose the correct statement(s):

(a) If X is an NP-complete problem, then X is an NP problem.

(b) If X is an NP-complete problem, then X is NP-hard.

(c) Let X be an NP-complete problem. If X can polynomially reduce to a problem Y, then Y is NP-complete.

(d) Let X be an NP-complete problem. If a problem Y can polynomially reduce to X, then Y is NP-complete.

(e) Let X be an NP-complete problem. If X can polynomially reduce to a problem Y, then Y is NP-hard.

18. Below is the Bellman-Ford Shortest Path (BFSP) algorithm, whose input is a positively weighted digraph, and whose output is the distance (or accumulated weight) of the shortest path from the source node to every node. Choose all items that all together make the algorithm a negative cycle detection algorithm, whose input is a weighted digraph *G* in which weights may be negative or positive, and whose output is " "yes" if graph *G* has cycles of negative weights; otherwise, "no" ".

(a) replace "$w[x][y]>0$" with "$w[x][y]>0$ or $w[x][y]<0$" in the **Input** section

(b) replace the **Output** with " "yes" if graph *G* has cycles of negative weights; otherwise, "no" "

(c) replace line 2 with

```
2. for $i \leftarrow 1$ to $|V|$ do
```

(d) replace line 6 with the following lines:

```
6. if $d[u]>0$ for each $u \neq s$ then
7.     return "yes"
8. else
9.     return "no"
```

(e) replace line 6 with the following lines:

```
6. for every edge $(x, y)$ in $E$ do
7.     if $d[y] > d[x] + w[x][y]$ then
8.         return "yes"
9. return "no"
```

參考用

Algorithm BFSP

Input: a weighted digraph *G*=(*V*, *E*), where the weight of each edge (*x*, *y*) in *E* is represented by

注意:背面有試題

$w[x][y]$, $w[x][y] \geq 0$, and a source node s, $s \in V$

Output: the distance of the shortest path from the source node to every node

```
1. d[s]←0;  d[u]←∞ for each u≠s
2. for i←1 to |V|-1 do
3.    for every edge (x, y) in E do
4.       if  d[y] > d[x] + w[x][y] then
5.          d[y]← d[x] + w[x][y]
6. return d
```

19. Below is Subset Sum Dynamic Programming (SSDP) algorithm to solve the subset sum problem, as described below. Given a set S of n positive integers, and a specific integer c, determine if there exists a subset S' of S such that the sum of elements in S' is c. Choose the correct statement(s).
(a). The time complexity of the SSDP algorithm is $O(n \cdot c)$.
(b). The time complexity of the SSDP algorithm is $O(n \cdot c^2)$.
(c). The SSDP algorithm is a pseudo-polynomial time-complexity algorithm.
(d). The subset sum problem is a P problem.
(e). The subset sum problem is an NP-complete problem.

Algorithm SSDP
Input: a set $S=\{v_1,\ldots,v_n\}$ of n positive integers, and a positive integer c
Output: "yes" if there exists a subset S' of S such that the sum of elements in S' is c; otherwise, "no"

```
1:  for w ← 0 to c do
2:      v[0, w] = 0
3:  for i ← 1 to n do
4:      for w ← 0 to c do
5:          if v_i ≤ w then
6:              v[i, w] ← max(v[i-1, w], v_i + v[i-1, w-v_i])
7:          else
8:              v[i, w] ← v[i-1, w]
9:  if v[n, c]=c then
10:     return "yes"
11: else
12:     return "no"
```

Below is the algorithm AllPairCost which computes the shortest distances between all pairs of vertices i, j, where i ≠ j. Formally, given distances of edges in graph G, determine the shortest distances between all pairs of vertices in G. Note that the distance of an arbitrary edge in G is

注意:背面有試題

non-negative.

Algorithm AllPairCost
Input: a two dimensional array C, where C[i][j]>0 denotes the distance of directed edge (i, j) (i.e., the edge from vertex i to vertex j).
Output: a two dimensional array D, where D[i][j] denotes the shortest distance from vertex i to vertex j.

```
1:  int i, j, k;
2:  for ( i=0; i<n; i++ )
3:     for ( j=0; j<n; j++ )
4:          D[i][j] = C[i][j];
5:  for ( k=0; k<n; k++ )
6:     for ( i=0; i<n; j++ )
7:          for ( j=0; j<n; j++ )
8:               if (____(B7)____)
9:                    ____(B8)______;
```

20. Blank (B7) in the algorithm above should be (a) D[i][j] < D[i-1][k]+D[k][j-1] (b) D[i][j] < D[i][k]+D[k][j] (c) D[i][j] < D[i][k+1]+D[k+1][j] (d) D[i][j] < D[i][k] +D[j][k] (e) D[i][j] < D[i-1][j-1]
21. Blank (B8) in the algorithm above should be (a) D[i][j] = D[i-1][k]+D[k][j-1] (b) D[i][j] = D[i][k]+D[k][j] (c) D[i][j] = D[i][k+1]+D[k+1][j] (d) D[i][j] = D[i][k] +D[j][k] (e) D[i][j] = D[i-1][j-1]
22. Which of the following statements is true? (a) For two non-adjacent vertices u and v, C[u][v] should be 0 (b) Algorithm AllPairCost is applicable to graphs with cycles (c) Algorithm AllPairCost is applicable to undirected graphs (d) Algorithm AllPairCost is applicable to graphs with multiple edges (i.e., two or more edges are incident to the same two vertices) (e) All of the above

注意:背面有試題