國立中央大學106學年度碩士班考試入學試題

所別： 資工類　　　　　　　　　　　　　　　　　　　　共5頁　第1頁

科目： 資料結構與演算法

本科考試禁用計算器　　　　　　　　　　　　　　　　　*請在答案卷內作答

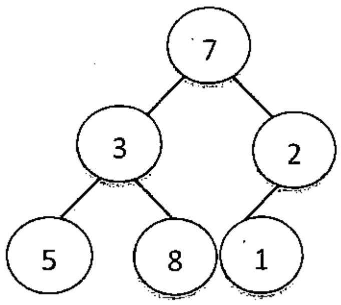
參考用

問答題

1. (5%) What is printed by the following algorithm?

```
A[ ]={0,1,2,3,4,5};
int min(int i, int j){
   int n; int m=(i+j)/2;
   if (i==j) return A[i];
   else {
       if (min(i, m) < min(m+1,j)) n=min(i, m);
       else n=min(m+1,j);
       printf("%d %d %d \n", i, j, n); return n; }
}
void main( ) {
    int L;
    L=min(0,5);
}
```

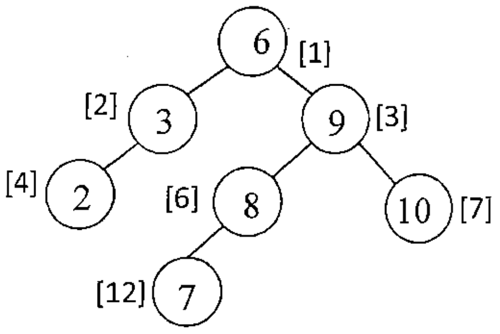
2. Consider the following binary tree



(a) (5 points) Adjust the binary tree above to establish a max heap. Draw the resulting max heap.
(b) (5 points) Draw the resulting max heap after inserting 4 into the max heap in (a).
(c) (5 points) Draw the resulting max heap after removing 8 from the max heap in (b).

注意：背面有試題

3. Consider the following binary search tree.



(a) (5 points) Show postorder traversal of the above binary search tree.
(b) (5 points) Please draw the resulting tree after inserting node 5 to the binary search tree above.
(c) (10 points) Consider a binary search tree stored in array A and each node's key value is a positive integer. For each tree node stored in A[*i*], its parent should be stored in A[*j*], where *j*=$\lfloor i/2 \rfloor$. For example, in the binary search tree above, the rightmost tree node (i.e., node 10) is stored in A[7] and its parent should be stored in A[3] (i.e., node 9). Fill the two blanks in the following procedure "preorder" (to complete a preorder traversal algorithm).

```
void main() {
    int A[];

    For (i=1; ...; i++)
        A[i] = −1. /*initial value*/
    preorder(1);
}

void preorder(int i) {
    If A[i]<> −1 {
        printf("%d", A[i])
        preorder(____________);
        preorder(____________);}
    }
```

所別： 資工類 共5頁 第3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 內作答

4. (10 points) Fill the following two blanks to complete an insertion sort algorithm (which sorts elements of array $A[1{:}N]$).

```
void insertionSort(int A[], int N)
{   int j, t;
    For (j=2; j<=N; j++){
          t = A[j];
          insert(t, A, j-1);}
}
void insert(int t, int A[], int i)
{
    A[0]=t;
    While (t<A[i]) {
       A[i+1] =__________ ;
       i--;}
    A[i+1] = _________;
}
```

5. Assume it is a known fact that problem X is NP-complete.

(a) How can we prove that problem Y is also NP-hard by taking advantage of polynomial-time reduction and the fact? (6%)

(b) If we can prove Y is NP-hard and Y is NP, then we can prove Y is also NP-complete. How can we prove that Y is NP? (6%)

6. The procedure of solving many problems can be represented by trees. Thus, the solving of these problems becomes a tree searching problem. There are many tree searching algorithms: breadth-first search, depth-first search, hill climbing and best-first search. Below is the pseudo code of the breadth-first search algorithm. Please modify the pseudo-code to be the depth-first search algorithm (6%), the hill climbing algorithm (6%) and the best-first search algorithm (6%). You should write down the complete algorithm including the input, the output and all steps.

Algorithm: Breadth-First Search Algorithm
Input: the root node r of a tree to search for the goal node g **Output**: the path from r to g as the solution or NIL to indicate failure to find a solution
Step 1: Construct a one-element queue consisting of the root node r Step 2: Check if the first element in the queue is the goal node g. If so, return the path from r to g as the solution and stop. Step 3: Remove the first element from the queue. Add all descendants of the first element, if any, to the end of the queue one by one. Step 4: If the queue is empty, then return NIL and stop. Otherwise, go to Step 2.

注意：背面有試題

所別： 資工類 共5頁 第4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 內作答

7. Below is the famous Dijkstra Shortest Path Algorithm (DSPA). The input of DSPA is a weighted directed graph $G=(V, E)$ and a specific source node s, where V is the node set and E is the edge set and every edge in E has a positive weight. (Note that the weight of the edge (u, v) is stored in $ew[u, v]$). For every node u in $V-\{s\}$, DSPA can derive the shortest path from s to u. However, DSPA does not consider node weights. If we now consider node weight and include node weights in the total weight of every path, we need to extend the original DSAP. Please extend DSAP to consider the node weights. Note that we assume the weight of a node u is stored in $nw[u]$ and the weight of the destination node is not included in the total weight of the path. Note that DSPA uses indentation to represent the block structure. So please use indentation properly when writing down the extended DSPA algorithm. You should write down the complete algorithm including the input, the output and all steps. (10%)

```
Algorithm: DSPA (Dijkstra Shortest Path Algorithm)
Input: G=(V, E), ew, s //G =(V, E) is a graph with edge weights stored in ew, and s is the source
Output: SP //SP is the set of shortest paths from s to all other nodes
1: dist[s]←0; dist[u]←∞, for each u≠s, u∈V
2: insert u with key dist[u] into priority queue Q, for each u∈V
3: while (Q≠∅)
4:     u←Extract-Min(Q)
5:     for each v adjacent to u
6:         if dist[v] > dist[u]+ew[u,v] then
7:             dist[v]←dist[u]+ew[u,v]
8:             pred[v]←u   //v's predecessor in the shortest path is u
9: calculate the shortest path from s to u to add into set SP according to pred[u], for each u∈V, u≠s
10: return SP
```

注意：背面有試題

國立中央大學106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

*請在答案卷內作答

8. Given a knapsack with capacity C, and n objects $o_1,\ldots,o_n$ with weights $w_1,\ldots,w_n$ and values $v_1,\ldots,v_n$, the 0/1 knapsack problem is to determine x_i ($x_i = 0$ or 1, $1 \le i \le n$) such that $\sum_{1\le i\le n} x_i w_i \le C$ and $V = \sum_{1\le i\le n} x_i v_i$ is maximized. Below is a dynamic programming algorithm to solve the 0/1 knapsack problem to output V. Please modify the algorithm to solve the subset sum problem, described as follows. Given a set S of n values $v_1, v_2, \ldots, v_n$ and a special value C, the subset sum problem is to determine whether or not there exists a subset S' of S such that $C = \sum_{v_i \in S'} v_i$. You should write down the complete algorithm including the input, the output and all steps. (10%)

Algorithm: 0/1 knapsack dynamic programming algorithm

Input: knapsack capacity C, n object weights $w_1,\ldots,w_n$ and n object values $v_1,\ldots,v_n$
Output: the maximum value V of all objects that can be kept in the knapsack

```
for $w \leftarrow 0$ to $C$ do
    $v[0, w] \leftarrow 0$
for $i \leftarrow 1$ to $n$ do
    for $w \leftarrow 0$ to $C$ do
        if $w_i \le w$ then
            $v[i, w] \leftarrow \max(v[i-1, w], v_i + v[i-1, w-w_i])$
        else
            $v[i, w] \leftarrow v[i-1, w]$
$V \leftarrow v[n, C]$
return $V$
```