國立成功大學

115學年度碩士班招生考試試題

編　號：139

系　所：電機資訊學院-資訊聯招

科　目：程式設計

日　期：0203

節　次：第 2 節

注　意：1.不可使用計算機
2.請於答案卷(卡)作答，於試題上作答，不予計分。

Part I. 資料結構 (50%)

注意: 資料結構共有 15 題。

一、問答題 (14 分): 本大題之答案請依照題號(1, 2, 3, 4)依序作答於答案卷。

1. [3%] A campus system stores device IDs. Each ID is a 6-bit binary string. Given the following seven 6-bit keys: 000001, 000010, 000011, 001111, 100000, 101011, 111110. Please insert the 7 keys into an empty digital search tree in the order given above. When searching for key 001111, list the keys visited on the search path, in order.
   Answer format: key1->key2->...
2. [4%] (Continue Question 1) Build the compressed binary trie for the same seven 6-bit keys. For each branch node, define its *bitNumber* as the bit index used at that node to choose the left or right child. The leftmost bit is index 1 and the rightmost bit is index 6. Perform BFS starting from the root. When visiting a branch node, enqueue the left child before the right child. List the *bitNumber* of each visited branch node in BFS order.
   Answer format: bitNum1, bitNum2, ...
3. [3%] A binary tree stores distinct keys. Based on the following postorder and inorder sequences, please list the keys of nodes at level 3 of the tree from left to right. Note that the root is at level 1.
   - Postorder sequence: A B C K D I G J H F E
   - Inorder sequence: A C B D K E I G F H J

   Answer format: key1, key2, ...
4. [4%] For the number of nodes $n = 15$, what is the maximum possible height of an AVL tree, and what is the maximum possible height of a red-black tree? Note that the root is at level 1.
   Answer format: AVL max possible height=?, Red-black tree max possible height =?

二、單選題 (24 分，每題 3 分): 本大題之答案請劃記於答案卡。共有題組 A 和題組 B，每題組包含 4 小題。每小題答對得 3 分，若答錯將倒扣 1 分。

A. A firewall uses a fast pre-check before running an exact check. All input integers are non-negative. You must answer **Questions** 1-4 based on the program below.

```
static inline uint32_t f1(int x) { return (uint32_t)(x % 8); }
static inline uint32_t f2(int x) { return (uint32_t)((x / 2) % 8); }
```

```
static inline void set1(uint32_t *b, uint32_t p) { *b |= (1u << p); }
static inline int  get1(uint32_t b, uint32_t p) { return (b >> p) & 1u; }
static int precheck(uint32_t bits, int x) {
    return get1(bits, f1(x)) && get1(bits, f2(x));
}
static int exact_check(int x) {
    return (x == 10) || (x == 22);
}
int main(void) {
    int base[] = {10, 22};
    uint32_t bits = 0;
    for (int i = 0; i < (int)(sizeof(base)/sizeof(base[0])); i++) {
        set1(&bits, f1(base[i]));
        set1(&bits, f2(base[i]));
    }
    int n;
    scanf("%d", &n);
    int exact_checks = 0, blocked = 0;
    for (int i = 0; i < n; i++) {
        int x;
        scanf("%d", &x);
        if (precheck(bits, x)) {
            exact_checks++;
            if (exact_check(x)) blocked++;
        }
    }
    printf("%d %d\n", exact_checks, blocked);
    return 0;
}
```

1. [3%] What is `precheck(bits, x)` mainly used for?

(A) It is an exact membership test because checking two bit positions avoids collisions.

(B) It is a probabilistic membership test that may have false positives.

(C) It is a bitmask-based two-condition rule: it passes a value if either of the tested bit positions is 1.

(D) It is a compact counter test: the function treats the two bit checks as a "count" and passes `x` only if the count is at least 2.

(E) It is a direct index check: `f1(x)` and `f2(x)` map each value to two unique bit positions.

2. [3%] What is the program output for the following input?

Input:

```
5
10 14 22 6 30
```

(A) 2 2
(B) 2 1
(C) 3 2
(D) 3 1
(E) None of the above

3. [3%] For the current program (`base = {10, 22}`), what is the smallest integer x such that `precheck(bits, x)` returns 1 and `exact_check(x)` returns 0?
(A) 4
(B) 5
(C) 6
(D) 7
(E) 11

4. [3%] Assume the blacklist policy of the firewall changes and the blacklist becomes `{10, 14, 22}`. To correctly apply the new policy, which parts of the program must be updated?
(A) `base` and `exact_check`
(B) `exact_check` only
(C) `precheck` only
(D) `precheck` and `base`
(E) No change is needed.

B. A university plans to connect all campus buildings by fiber links. Each building is a vertex (ID = 0, 1, ..., n-1), and each possible fiber link is an undirected edge `(a,b)` with installation cost w. Assume the cost w is an integer and satisfies $0 \le w \le 10^9$. The following C program is designed to find the minimum cost of installing connections from the given links. When two edges have the same cost, the program orders them by `(a,b)` in increasing order (smaller a first; if a is the same, smaller b first). You must answer **Questions 5-8** based on the C program below.

```
typedef struct { int a, b, w; } E;
static int *P, *S;
static void f0(int n){
    P = (int*)malloc(n * sizeof(int));
    S = (int*)malloc(n * sizeof(int));
    for(int i=0;i<n;i++){ P[i]=i; S[i]=1; }
}
static int f1(int x){
    while(P[x]!=x){ P[x]=P[P[x]]; x=P[x]; }
    return x;
}
static int f2(int x, int y){
    x=f1(x); y=f1(y);
```

```
        if(x==y) return 0;
        if(S[x]<S[y]){ int t=x; x=y; y=t; }
        P[y]=x; S[x]+=S[y];
        return 1;
    }
    static int c0(const void *px, const void *py){
        const E *x=(const E*)px, *y=(const E*)py;
        if(x->w != y->w) return x->w - y->w;
        if(x->a != y->a) return x->a - y->a;
        return x->b - y->b;
    }
    int main(void){
        int n, m;
        if(scanf("%d %d", &n, &m)!=2) return 0;

        E *e = (E*)malloc(m * sizeof(E));
        for(int i=0;i<m;i++){
            scanf("%d %d %d", &e[i].a, &e[i].b, &e[i].w);
            if(e[i].a > e[i].b){
                int t=e[i].a; e[i].a=e[i].b; e[i].b=t;
            }
        }
        qsort(e, m, sizeof(E), c0);
        f0(n);
        int *U = (int*)malloc((n-1) * sizeof(int));
        int *V = (int*)malloc((n-1) * sizeof(int));
        int k=0;
        long long total=0;
        for(int i=0;i<m && k<n-1;i++){
            if(f2(e[i].a, e[i].b)){
                U[k]=e[i].a;
                V[k]=e[i].b;
                total += e[i].w;
                k++;
            }
        }
        printf("%lld %d\n", total, k);
        free(e); free(U); free(V); free(P); free(S);
        return 0;
}
```

5. [3%] Which description best matches what the program does to build the backbone?

(A) It starts from one vertex and repeatedly adds the cheapest edge that leaves the visited vertices.

(B) It sorts all edges by cost, scans them in that order, and adds an edge only if it connects two different groups.

(C) It visits vertices level by level using a queue and records a tree of visits.

(D) It repeatedly selects the closest vertex based on a distance array and updates distances.

(E) It computes shortest paths between every pair of vertices, then selects edges from those paths.

6. [3%] What is the program output for the following input?

Input:

```
6 11
0 1 4
0 2 3
0 3 6
1 2 5
1 4 2
2 3 7
2 5 8
3 5 1
4 5 9
1 3 10
0 4 11
```

(A) `16 5`

(B) `16 4`

(C) `15 5`

(D) `15 4`

(E) `14 4`

7. [3%] The backbone is the graph formed by the `k` chosen edges. A vertex is an articulation point if removing it (and its incident backbone edges) disconnects the backbone. For the dataset in Question 6, which set of vertices are articulation points in the backbone?

(A) {0, 1, 4}

(B) {0, 2, 3}

(C) {1, 3, 5}

(D) {0, 1, 3}

(E) No articulation point

8. [3%] The operator wants one more feature: besides the current output, also print the number of connected components in the input graph (using all `m` edges). Print three integers in one line:

```
total k c
```

where `c` is the number of connected components. Which modification is correct?

(A) Add the following code right before `printf(...)`, and change the printf format:

```
int c = 0;
```

```
for(int i=0;i<n;i++){
    if(f1(i) == i) c++;
}
printf("%lld %d %d\n", total, k, c);
```

(B) Add the following code right before `printf(...)`, and change the printf format:

```
int c = 0;
for(int i=0;i<n;i++){
    if(P[i] == -1) c++;
}
printf("%lld %d %d\n", total, k, c);
```

(C) Add the following code right before `printf(...)`, and change the printf format:

```
int c = 0;
for(int i=0;i<n;i++){
    if(S[i] == 1) c++;
}
printf("%lld %d %d\n", total, k, c);
```

(D) Add the following code right before `printf(...)`, and change the printf format:

```
int c = 0;
for(int i=0;i<n;i++){
    c += (P[i] == i);
}
printf("%lld %d %d\n", total, k, c);
```

(E) None of the above.

三、複選題 (12 分，每題 4 分): 本大題之答案請劃記於答案卡。每題答對得 4 分。

9. [4%] You need operations: `insert`(x), `delete`(x), `find`(x), and also "`find predecessor/successor of` x". Please select all that are correct for the predecessor/successor queries.

(A) A red-black tree supports the queries in O(log n) worst case when the input is a key x.

(B) A binary heap supports the queries in O(log n) worst case when the input is a key x.

(C) A B+ tree supports the queries in O(1) average time when the input is a key x.

(D) An AVL tree supports the queries in O(log n) worst case when the input is a key x.

(E) Hashing supports the queries in O(1) average time when the input is a key x.

10. [4%] A system needs a priority queue with: frequent `meld(A, B)`, frequent `delete-min`, and occasional `find-max`. Please select all that are correct.

(A) A Fibonacci heap is a good match for frequent meld and delete-min.

(B) A leftist heap is a good match for frequent meld and delete-min.

(C) An SMMH directly supports find-max and find-min efficiently.

(D) A Fibonacci heap supports find-max in O(1) without any extra structure.

(E) A leftist heap supports find-max in O(1) without extra structure.

11. [4%] A database index is stored on disk. Minimizing disk I/O is important. Fan-out means the maximum number of children of a node. Please select all that are correct.

(A) A B-tree is suitable for disk because a node can store many keys and child pointers in one disk page/block. This gives high fan-out and small height (few page reads).

(B) A red-black tree is suitable for disk because its fan-out is 2, which reduces the number of disk page reads.

(C) A B+ tree is suitable for disk indexes for the same reason as a B-tree (high fan-out, small height).

(D) A B+ tree is often used when range queries are common, because its leaves can be linked for fast sequential scan on disk.

(E) For disk-based indexes, a red-black tree usually needs more page reads than a B-tree/B+ tree when searching for a key, because the tree is taller.

Part II. 演算法 (50%)

注意: 演算法共有 9 題

一、是非題 (10 分，每題 2 分，答錯倒扣 1 分): 請將您的答案劃記於在答案卡，否則，不予計分。

12. [2%] Consider an algorithm with a running time function $T(n) = 50n^2 + 200n\log n + 10^6$. Is it mathematically correct to claim that $T(n) = O(n^2)$?

(A) True (B) False

13. [2%] Consider the famous "Halting Problem", which has been mathematically proven to be undecidable. Because the Halting Problem is strictly harder than any problem in NP, it satisfies the definition of being NP-Hard. Since the Halting Problem satisfies the NP-Hard property, it is therefore correctly classified as an NP-Complete (NPC) problem.

(A) True (B) False

14. [2%] If a decision problem belongs to the class NP, it implies that given a specific candidate solution, there exists a deterministic algorithm that can verify whether this solution is correct in polynomial time.

(A) True (B) False

15. [2%] The 0/1 Knapsack Problem can be solved optimally by using a Greedy strategy that always selects the item with the highest value-to-weight ratio first.

(A) True (B) False

16. [2%] Suppose a problem called "Robot-Path-Planning" is a known NP-Complete problem. To prove that a related new problem "3-SAT" is also NP-Complete, we must demonstrate that "3-SAT" is in NP, and we must construct a polynomial-time reduction from "3-SAT" to "Robot-Path-Planning"

(A) True (B) False

二、問答題 (40 分)：本大題之答案請作答於答案卷，並請註明該答案所對應之題號 (例: 1-1、 1-3.2)

1. [10%] Assuming that you are a medical information analyst, currently analyzing an algorithm designed to process a 3D MRI volume. The input is a cubic volume of data with side length n (total voxels = $n \times n \times n$). Assume n is a power of 2.

Pseudo-code:

```
Surface_Scan(n)
    total_intensity = 0
    for i = 1 to n
        for j = 1 to 100
            total_intensity = total_intensity + Read_Pixel(i, j)
    return total_intensity

MRI_Voxel_Process(n)
    if n <= 1
        return

    for k = 1 to n
        Surface_Scan(n)

    for x = 0 to 1
        for y = 0 to 1
            for z = 0 to 1
                MRI_Voxel_Process(n/2)
```

Note: Assume that the function Read_Pixel(i, j) takes constant time

Questions:

(1) [3%] Analyze the **Surface_Scan(n)** function and determine its time complexity using Θ-notation.

- Time complexity: _____

(2) [3%] Write down the recurrence relation for the running time T(n) of **MRI_Voxel_Process(n)**.

- T(n) = _____

(3) [4%] Analyze the asymptotic growth rate of the algorithm, and determine the overall asymptotic tight bound (Θ) for the running time of **MRI_Voxel_Process(n)**. Please write down your calculation process.

- Calculation process: _____
- Overall asymptotic tight bound: _____

2. [10%] You are a process engineer managing a critical Lithography Machine. The machine has a strict **Thermal Constraint**:

Constraint: You cannot schedule production for two consecutive shifts. If Shift i is active, Shift $i-1$ and Shift $i+1$ must be idle (cooling).

Data: The estimated Production Yield Value for the upcoming 13 shifts is listed below.

Shift ID	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	S13
Value	45	95	45	35	85	45	40	85	65	65	40	35	90

Questions:

(1) [4%] Identify the set of shifts that results in the maximum possible total value while satisfying the thermal constraint.

- Selected Shifts: _____ (e.g., S1, S2, …)
- Total Value: _____

(2) [4%] A very important client demands that **Shift 7 (Value: 40) MUST** be included in the schedule. Under this mandatory condition, what is the maximum total value you can achieve for the 13 shifts?

- Selected Shifts with S7: _____
- Total Value with S7: _____

(3) [2%] **Process Upgrade**: The High-Temperature Machine Imagine the factory installs a new, high-performance machine. This machine generates significantly more heat, so the safety constraint is updated:

New Constraint: After any active shift, the machine must cool down for **TWO** consecutive shifts

Let $f[i]$ be the maximum value achievable considering shifts up to i, and let v_i be the value of Shift i. Please write down the complete Recurrence Relation (Mathematical Equation) for

- $f[i]$: _____

3. [10%] You are analyzing the routing latencies for a constellation of 4 satellites (Nodes A, B, C, D). Below is the adjacency list with the latency between specific nodes:

Adjacency List:

- A → B: 10
- A → C: -5
- C → B: 4
- B → D: 5
- D → C: 2
- D → A: 3

Requirement: You need to analyze the latency characteristics of the network so that shortest communication paths can be efficiently determined when needed

Questions:

(1) [4%] Based on the specific properties of the edges in this graph and the problem requirement, identify the **single most appropriate standard algorithm** to use. **Explain your choice** by pointing out the specific graph feature that disqualifies other algorithms, and explicitly state **why** at least one other commonly used shortest-path algorithm would fail or be inappropriate in this case.

- Algorithm: _____
- Why choosing this algorithm? _____

(2) [2%] Let M be the initial 4×4 **distance matrix** used at the start of your chosen algorithm. Please construct and write down M clearly.
(Note: Use "∞" for non-existing edges and "0" for self-loops. Order: A, B, C, D)

- $M = \begin{bmatrix} & \\ & \end{bmatrix}$

(3) [4%] After the algorithm completes execution

(a) [2%] What is the **final shortest path cost** from Node D to Node B? Please **trace the path logic** (e.g., D → … → B).

- Final shortest path cost: _____
- Final path logic: _____

(b) [2%] State the asymptotic **Time Complexity** Θ of this algorithm.

- Time Complexity: _____

4. [10%] You are optimizing the core engine of an image editing tool called "Smart Lasso". The tool helps users extract objects by automatically finding the best boundary between a **Start Pixel (S)** and a **Target Pixel (T)**.
The image is represented as a grid of pixels. Each pixel has an associated "Energy Cost" (representing high-frequency noise or weak edges). Here are some rules:

- **Movement Rule**: From any pixel, you can move to adjacent pixels (Up, Down, Left, Right). Diagonal movement is not allowed.
- **Path Cost**: The sum of the Energy Costs of all pixels visited on the path (excluding S, but including T).

Energy cost matrix is shown as below (assume the cost of entering T is 0):

$$\begin{bmatrix} S & 4 & 5 & 9 & 8 \\ 7 & 9 & 8 & 6 & 9 \\ 3 & 2 & 4 & 1 & 5 \\ 6 & 5 & 6 & 7 & T \end{bmatrix}$$

In the following, we will use (row, col) as coordinate representation. Top-left is (0, 0).

If two neighbors have the same cost, prioritize the direction in this order:
Right → Down → Up → Left

Questions:

(1) [4%] A junior developer implemented a "High-Speed Mode" for the lasso. The logic is simple:

- "At every step, the cursor simply snaps to the immediate neighbor with the lowest energy cost to get closer to the target. It does not plan ahead."

Please **trace the path** generated by this method from **S** to **T** and **calculate the total cost.**

- Path Sequence: _____ (e.g., (0,0) → (0,1) → ...)
- Total Path Cost: _____

(2) [4%] Users have complained that the "High-Speed Mode" (from Q 4.1) often gets "trapped" in noisy areas or takes expensive detours, failing to find the true object boundary. Your manager asks you to rewrite the core engine. The new requirement is:

- "The tool must guarantee finding the mathematically optimal boundary—the path with the absolute minimum Total Cumulative Cost from Start to Target—even if it takes a bit more computation time."

Please **trace the path** generated by this method from **S** to **T** and **calculate the total cost.**

- Path Sequence: _____ (e.g., (0,0) → (0,1) → ...)
- Total Path Cost: _____

(3) [2%] The solution in Q 4.2 works great on small grids. However, when applied to a 4K Image (with **N** pixels), the tool becomes laggy. Profiling reveals the bottleneck is in the **Open Set (Priority Queue)**—specifically, the operation of repeatedly finding and removing the pixel with the minimum cumulative cost. You are **comparing two underlying data structures** for this Priority Queue: a simple **Unsorted Array** vs. a **Binary Min-Heap**. Please analyze the Time Complexity (Big-O) for the "Extract-Min" operation in both cases:

- **Case 1**: You use a simple **Unsorted Array** to store the candidate pixels, the time complexity for Extract-Min is: _____
- **Case 2**: You switch to a **Binary Min-Heap** to store the candidate pixels, the time complexity for Extract-Min is: _____