# 國立交通大學105學年度碩士班考試入學試題

科目：資料結構與演算法(1001)　　考試日期:105年2月3日 第 1 節

系所班別：資訊聯招　　第 1 頁,共 11 頁

【不可使用計算機】＊作答前請先核對試題、答案卷(試卷)與准考證之所組別與考科是否相符！！

注意：每大題有三小題，每小題必須全對該大題才給分，每大題五分。 請使用答案卡作答

**1.** A certain binary tree has the following traversal sequences (each letter represents a tree node):
Inorder: **FABEGHCD**; postorder: **FAEHCDGB**.

(1) Which of the following is TRUE about the ordering of nodes in the preorder traversal sequence?
(A) **G** is before **A** (B) **H** is before **E** (C) **E** is before **B** (D) **C** is before **D** (E) **A** is before **F**.

(2) Which node appears at the fifth position in the level-order traversal (from left to right) sequence?
(A) **E** (B) **F** (C) **B** (D) **D** (E) **G**

(3) What is the height of this tree? Here we assume the lowest leaf node has height 1.
(A) 3 (B) 4 (C) 5 (D) 6 (E) 7

**2.** Consider the conversion of the following infix expression to its postfix form: **a/(b+c*d)*(e–f)+g**

(4) Consider the stack used for this conversion. What is the maximum number of tokens in this stack at any moment during the conversion?
(A) 1 (B) 2 (C) 3 (D) 4 (E) 5

(5) The first operator that appears in the postfix expression is
(A) **(** (B) ***** (C) **/** (D) **+** (E) **–**

(6) In the postfix expression, which of these tokens is an operand?
(A) the last token (B) the eighth token (C) the sixth token (D) the tenth token (E) all of the above

**3.** Consider the following code of a singly linked list, with the function **F** to "rotate" the list by moving the first item of the list to the end of the list. The input to the function is the address of **first**, the pointer to the first item in the list. You need to select the correct expressions to put into the code in the following three questions so that the function works correctly.

```
struct node { int data;  struct node *next; };
void F(struct node **pfirst) {
  struct node *x, *y, *first = *pfirst;
  if (first && first->next) {
    x = first;
    y = *pfirst = ____①____;
    while (y->next) y = y->next;
    ____②____ = x;
    ____③____ = NULL;
  }
}
```

(7) What expression should be in ①?
(A) **x** (B) **x->next** (C) **y->next** (D) **first** (E) **x->next->next**

(8) What expression should be in ②?
(A) **y** (B) **x->next** (C) **y->next** (D) **first** (E) ***pfirst**

(9) What expression should be in ③?
(A) **x** (B) **x->next** (C) **y->next** (D) **first** (E) ***pfirst**

**4.** A binary search tree is constructed by inserting nodes with the following keys into an initially empty tree, in the given order: 10, 2, 4, 6, 8, 1, 3, 5, 12. The nodes are represented by their keys in the questions.

(10) Which of these nodes is at level 3? (The root is at level 1.)
(A) 6 (B) 12 (C) 3 (D) 5 (E) 1

(11) How many leaf nodes are there in this tree?
(A) 4 (B) 2 (C) 3 (D) 6 (E) 5

(12) Let us delete node 4, and then delete node 2. Which of these relations between nodes 3 and 6 is impossible in the resulting tree?
(A) 3 is the parent of 6, (B) 3 is the grandparent of 6, (C) 3 is the sibling of 6, (D) 3 is a child of 6, (E) none of the above.

**5.** A red-black tree is constructed by inserting nodes with the following keys into an initially empty tree, in the given order: 10, 2, 4, 6, 8, 1, 3, 5, 7, 9. The nodes are represented by their keys in the following questions.
(13) Which of the following node pairs contains two black nodes?
(A) 2 and 6 (B) 4 and 8 (C) 7 and 10 (D) 3 and 10 (E) 4 and 5
(14) How many red nodes are there in this tree?
(A) 3 (B) 4 (C) 5 (D) 6 (E) 7
(15) Which node is the root?
(A) 4 (B) 5 (C) 6 (D) 8 (E) 10

**6.** This question is about Array:
(16) Which of the following 3 statements (a), (b) and (c) is(are) true?

(a)C++ code,

```
char *A;
A=new char [1000];
cout <<A[10] and cout << *(A+10) get the same output.
```

(b)Array supports random access, an array A of length $n$, accessing A[1000] needs more time than to accessing A[1].
(c) Array is the most fundamental data structure. We can use it to implement any data structure, even the user define structure.

(A) Only (a) is true.
(B) Only (b) is true.
(C) (a) and (b) are true, (c) is not.
(D) (a) and (c) are true, (b) is not.
(E) (b) and (c) are true, (a) is not.

(17) A length $n$ vector (linear algebra vector) can be stored in an array. Which of the following statement(s) is(are) true?
(a) The operation of adding two vectors takes $\Theta(n)$ time and $\Theta(n)$ space.
(b) If the two vectors are sparse; there are $m$ out of $n$ non-zero terms. We can store the $m$ terms in an array of length $m$. Each entry stores the index and the value. The operation of adding two vectors takes $\Theta(m)$ time and $\Theta(n)$ space.
(c) The two sparse vectors as (b), we now store the two sparse vectors in the linked list. Each node has the index and the value. The operation of adding two vectors takes the same $\Theta(m)$ time and $\Theta(n)$ space. We prefer to use linked list representation because we don't have to worry about the number of non-zero terms that may not be $m$ after addition.

(A) Only (a) is true.
(B) Only (c) is true.
(C) (a) and (b) are true, (c) is not.
(D) (a) and (c) are true, (b) is not.
(E) (b) and (c) are true, (a) is not.

(18) C string is implemented using array. Given two string *pat* and *text*, string matching is to determine whether there is a matching of *pat* in *text*. Let the length of *pat* and *text* be respectively $m$ and $n$. Which of the following statement(s) is(are) true?

(a) A very efficient string matching algorithm needs to compute a failure function which is obtained from *pat* and *text*.
(b) Using the failure function, string matching can be done in $O(m+n)$ time.
(c) A naive approach without using the failure function takes $O(mn)$ time.

(A) Only (a) is true.
(B) Only (c) is true.
(C) (a) and (b) are true, (c) is not.
(D) (a) and (c) are true, (b) is not.
(E) (b) and (c) are true, (a) is not.

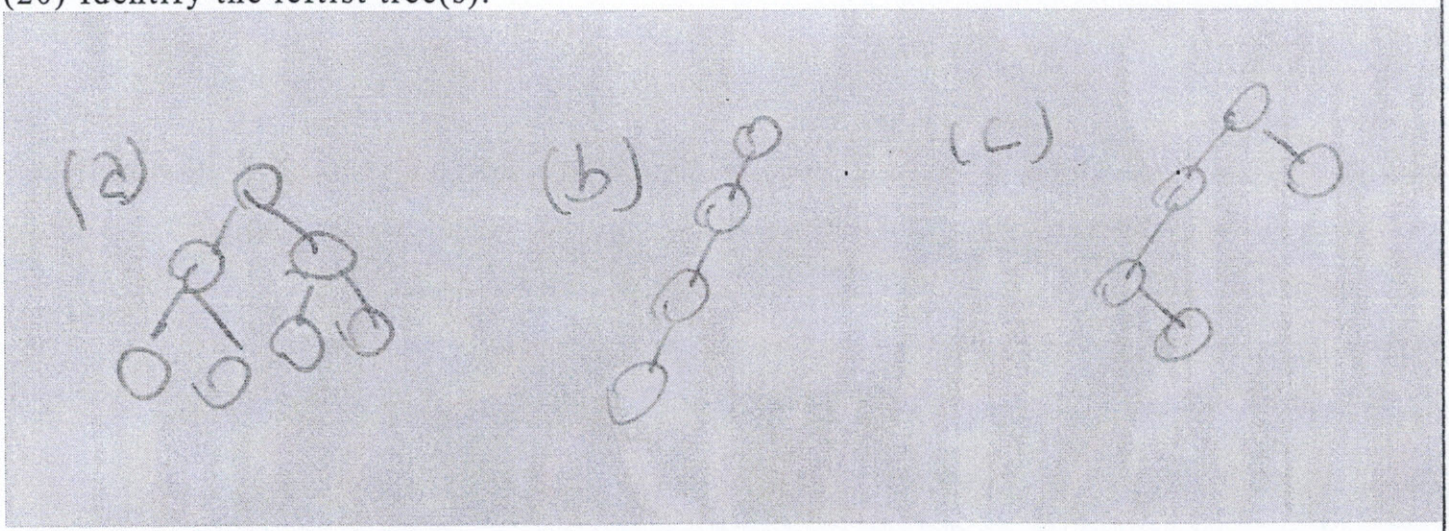
**7.** This question is about Priority Queue.

(19) About the binary heap, which of the following statement(s) is(are) true?

(a) Binary heap is a complete binary tree stored in an array, so we do not have to store the left-child and right-child pointers.
(b) Building a min-heap of size n takes $\Theta(n \log n)$ time.
(c) Merging 2 min-heaps takes $\Theta(n)$ time.

(A) (a) and (b) are true, (c) is not.
(B) (a) and (c) are true, (b) is not.
(C) (b) and (c) are true and (a) is not.
(D) all of (a), (b), and (c) are correct.
(E) only (a) is true.

(20) Identify the leftist tree(s).




(A) (a) and (b) are leftist tree, (c) is not.
(B) (b) and (c) are leftist trees, (a) is not.
(C) (a) and (c) are leftist tree, (b) is not.
(D) all of (a), (b), and (c) are leftist tree.
(E) Only (c) is leftist tree.

(21) Which of the following statement(s) is (are) true.

(a) The height of the leftist tree is $O(\log n)$.

(b) Using a min-leftist to implement priority queue, it supports $O(\log n)$ merging two min-heaps.

(c) Insert arbitrary and delete min are carried out by merging two min-heaps, so the time required is $O(\log n)$.

(A) All of (a), (b), and (c) are true.
(B) (a), (b) are true, (c) is not.
(C) (b), (c) are true, (a) is not.
(D) (a), (c) are true, (b) is not.
(E) only (a) is true.

**8.** This question is about Sorting:

(22) Which of the following 3 statement(s) is(are) true?

(a) The worst case of quick sort takes $O(n^2)$ time.

(b) The worst case of the randomized quick sort takes $O(n^2)$ time.

(c) Quick sort is divide and conquer approach. After partitioning, we don't spend time for the merging step.

(A), (a) and (b) are true, but (c) is not.
(B) (a) and (c) are true, but (b) is not.
(C) (b) and (c) are true, but (a) is not.
(D) all of (a), (b), and (c) are true.
(E) Only (a) is true.

(23) Implementation of a sorting algorithm, we can always make it "stable" by using some tricks. We assume that the records are stored in an array. If we want to sort the array into non-decreasing order, which of the statement(s) is(are) true?

(a) To the quick sort, each record has 2 keys. The original key is the primary key and the index of the record in the array is secondary key. A comparison starts with comparing the primary key, if primary key is the same, we compare the secondary key.

(b) To the merge sort, If we have same key during merging two arrays, the record in the first array is placed before the record in the second array.

(c) To the insertion sort, when we insert record $i$, we insert it after record $j$ where record $j$ is the first one whose key is less than the key of record $i$.

(A) (a) and (b) are true, but (c) is not.
(B) (a) and (c) are true, but (b) is not.
(C) (b) and (c) are true, but (a) is not.
(D) all of (a), (b), and (c) are true.
(E) Only (a) is true.

(24) About the lower bound to the sorting problem, which of the following 4 statement(s) is(are) true?

(a) Heap sort needs at least $n \log n$ comparisons and heap sort can be done in at most $n \log n$ operations, heap sort is optimal.

(b) Merge sort is divide and conquer approach and the time required is modeld by $T(n) = 2T(n/2) + \Theta(n)$. Solving $T(n)$, we get $\Theta(n \log n)$. Thus merge sort needs at least $n \log n$ time.

(c) Lower bound for sorting algorithms using comparison operation established under the linear decision tree model.

(d) Sorting algorithms using non-linear operators have chance to beat the $\Theta(n \log n)$ lower bound.

(A) (a) (b) (c) are true, but (d) is not.
(B) (a) (b) (d) are true, but (c) is not.
(C) (a) (c) (d) are true but (b) is not.
(D) (b) (c) (d) are true but (a) is not.
(E) All of (a) (b) (c) and (d) are true.

**9.** Consider an undirected graph $G=(V,E)$, an edge is a relation over the vertices; the relation is reflexive, symmetric, and transitive. Computing the connected components in the $G$ is the same as to compute the equivalence classes of the vertices, $V$. Let the number of vertices be $n$ and the number of edges be $m$.

(25) Which of the following statement(s) is(are) true?

(a) Both breadth first search and Union/Find algorithm can compute the equivalence classes.

(b) Both depth first search and breadth first search can compute the equivalence classes. Performance of depth first search is better than the breadth first search.

(c) If graph $G$ is big (large $m$ and large $n$), among the breadth first search, depth first search, and Union/Find, we should choose depth first search because of space consideration.

(A) (a) and (b) are true, but (c) is not.
(B) (a) and (c) are true, but (b) is not.
(C) (b) and (c) are true, but (a) is not.
(D) all of (a), (b), and (c) are true.
(E) Only (a) is true.

(26) Union/Find implements set operations by using forest. A set is a rooted tree where the root presents the set. Recall that there are $n$ vertices and $m$ edges. About computing the equivalence classes, which of the statement(s) is (are) true.

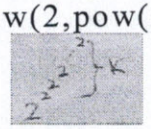
(a) Union operation can be done in $\Theta(1)$ time, there are at most $m$ Unions.

(b) The time required by Find($i$) operation depends on the path length from $i$ to the root. There are at most $m$ Finds.

(c) Applying weighting rule for Union, the time required for a Find is bounded above by $O(\log n)$. Applying collapsing rule for Find, the path length can be further reduced.

(A) (a) and (b) are true, but (c) is not.
(B) (a) and (c) are true, but (b) is not.
(C) (b) and (c) are true, but (a) is not.
(D) all of (a), (b), and (c) are true.
(E) Only (a) is true.

(27) Function $f_1(k)=2^k$, $f_1$ is the exponential function.
Function $f_2(k)=\text{pow}(2,\text{pow}(2,\text{pow}(2, \ldots, \text{pow}(2,2))))$, a tower of power of 2, and the height of the tower is $k$.



If $n= f_1(k)$, then $k$=log $n$. If $n= f_2(k)$, we say $k$=log*$n$. Which of the following statement(s) is(are) true?

(a) $f_2(k)$ grows faster than $f_1(k)$.

(b) log*$n$ grows faster than log $n$.

(c) According the speed of growth of the functions, we have $n$ log$n$ > $n$log*$n$ > $n$^α, where 1<α<1.01and $n$^α=pow(n, α).

(A) (a) and (b) are true, but (c) is not.

(B) (a) and (c) are true, but (b) is not.

(C) (b) and (c) are true, but (a) is not.

(D) all of (a), (b), and (c) are true.

(E) Only (a) is true.

**10.** About amortized cost analysis and some related algorithms.

(28) Which of the following statement(s) is (are) true?

(a) A sequence of operations, some of those can be done very fast but some can not. Amortized cost is to take the average of the costs of the sequence of operations. Similarly to the analysis of the quick sort, it is average case analysis and we estimate the expected cost.

(b) Amortized cost can be calculated by the accounting method. We pre-pay some cost. If an operation costs more, it can be covered by the pre-paid. All we have to make sure is that the pre-paid is always positive.

(c) STL in C++, the vector is implemented by using dynamic table. When an insertion into a full table, the vector size is double, all entries in the old table are copied into the new table, then the insertion is made. A sequence of n insertion takes linear time because a single insertion has amortized cost O(1).

(A) (a) and (b) are true, but (c) is not.

(B) (a) and (c) are true, but (b) is not.

(C) (b) and (c) are true, but (a) is not.

(D) all of (a), (b), and (c) are true.

(E) Only (a) is true.

(29) About heap, which of the following statement(s) is (are) true?

(a) Fibonacci heap support efficient deleting minimum, inserting arbitrary, and merging of two min-heaps.

(b) Fibonacci heap supports delete minimum in $O$(log $n$) amortized time.

(c) Fibonacci heap supports constant ($O$(1)) amortized cost for merging and inserting arbitrarily.

(A) (a) and (b) are true, but (c) is not.

(B) (a) and (c) are true, but (b) is not.

(C) (b) and (c) are true, but (a) is not.

(D) all of (a), (b), and (c) are true.

(E) Only (a) is true.

(30) Queue operations can be implemented using 2 stacks, S1 and S2.

```
inQueue (e) == S1.push(e);
dQueue() == if (S2.empty()) then{
                if (S1.empty()) the queue is empty;
                else pop out everything in S1 and push
                     them into S1;
                return S1.pop();
```

}

If we have a sequence of $n$ inQueue and dQueue, which of the following statement(s) is(are) true?

(a) inQueue(e) can be done in constant time.

(b) A single dQueue() takes $O(n)$ time because we may need to pop out $O(n)$ items in the S2.

(c) The sequence of $n$ inter-mix inQueue and dQueue operations takes $\Theta(n)$ time.

(A) (a) and (b) are true, but (c) is not.
(B) (a) and (c) are true, but (b) is not.
(C) (b) and (c) are true, but (a) is not.
(D) all of (a), (b), and (c) are true.
(E) Only (a) is true.

**11.** Given a hash table of size $m$, please answer following questions.

(31) Under the uniform hashing assumption, if we use the hash table with open addressing to hash 3 keys, the probability that the third inserted key needs exactly three probes before being inserted into the table is exactly
(A) $2/m$ (B) $2/(m-1)$ (C) $2/m(m-1)$ (D) $1/(m-1)$ (E) $1/m(m-1)$

(32) We use the hash table with open addressing to hash $n$ keys, where $n$ is less than $m$. Under the uniform hashing assumption, the expected cost to insert another element into the table is at most
(A) $1+ n/m$ (B) $1/(1 - n/m)$ (C) $n/m$ (D) $1/(n/m)$ (E) $1- n/m$

(33) After hashing $n$ keys into the hash table that uses chaining to handle collisions, we hash two new keys $k_1$ and $k_2$. Under the simple uniform hashing assumption, the probability that $k_1$ and $k_2$ are hashed into the same table location is exactly
(A) $1/n$ (B) $1/m$ (C) $1/nm$ (D) $n/m$ (E) $m/n$

**12.** Solve the recurrences in (33) to (35). Whenever possible, apply the Master Theorem. If the Master Theorem does not apply, use a recursion tree or substitution method to determine your answer.

(34) $T(n) = 7T(n/2) + n^3$
(A) $\Theta(n)$ (B) $\Theta(\log n)$ (C) $\Theta(n^2)$ (D) $\Theta(n \log n)$ (E) $\Theta(n^3)$

(35) $T(n) = T(n - 2) + n$
(A) $\Theta(n)$ (B) $\Theta(\log n)$ (C) $\Theta(n^2)$ (D) $\Theta(n \log n)$ (E) $\Theta(n^3)$

(36) $T(n) = 8T(n/2) + n^2$
(A) $\Theta(n)$ (B) $\Theta(\log n)$ (C) $\Theta(n^2)$ (D) $\Theta(n \log n)$ (E) $\Theta(n^3)$

**13.** Consider the following algorithm. The algorithm FIND_MODE(A) has input array arr containing $n$ different integer values.

```
#include <stdio.h>

int findPeakUtil(int arr[], int low, int high, int n)
{
    int mid = low + (high - low)/2;

    if ((mid == 0 || arr[mid-1] <= arr[mid]) &&
            (mid == n-1 || arr[mid+1] <= arr[mid]))
        return mid;
```

```
    else if (mid > 0 && arr[mid-1] > arr[mid])
        return findPeakUtil(arr, low, (mid -1), n);

    else return findPeakUtil(arr, (mid + 1), high, n);
}

int FIND_MODE(int arr[], int n)
{
    return findPeakUtil(arr, 0, n-1, n);
}
```

(37) FIND_MODE employs what approach?
(A) divide and conquer (B) dynamic programming (C) greedy (D) heuristic (E) approximation

(38) What is the time complexity of FIND_MODE?
(A) $\Theta(\log n)$ (B) $\Theta(n)$ (C) $\Theta(n \log n)$ (D) $\Theta(n^2)$ (E) None of above

(39) Assume input array arr containing 1, 3, 20, 4, 1, 0, what is the output?
(A) 1 (B) 0 (C) 20 (D) 3 (E) 2

**14.** Calculate the minimum number of multiplications necessary to find the matrix product ABCDE. The dimensions of each of the matrices are listed below:

| Matrix | Dimensions |
|---|---|
| $A_1$ | 2x4 |
| $A_2$ | 4x3 |
| $A_3$ | 3x2 |
| $A_4$ | 2x5 |
| $A_5$ | 5x1 |

(40) What kind of algorithm you should apply to obtain an optimal solution?
(A) divide and conquer (B) dynamic programming (C) greedy (D) heuristic (E) approximation

(41) What is the minimum number of multiplications to compute matrix product ABCDE?
(A) 24 (B) 56 (C) 64 (D) 36 (E) 28

(42) What is the optimal parenthesization of $A_1\ A_2\ A_3\ A_4\ A_5$?
(A) $(A_1A_2)\ A_3\ (A_4\ A_5)$
(B) $(A_1\ (A_2\ A_3)\ (A_4\ A_5)$
(C) $A_1\ (A_2\ ((A_3\ A_4)\ A_5))$
(D) $A_1\ (A_2\ A_3)\ (A_4\ A_5)$
(E) None of above

**15.** Construct Huffman codes for the letter a, b, c, d, e, and f with corresponding to frequencies 6, 11, 7, 9, 18, and 2. Note that we label the left branch 0 and 1 for the right branch in the Huffman tree. For sibling nodes, the frequency of the left node is at most the frequency of the right node.

(43) Huffman invented what kind of algorithm that constructs a Huffman code?
(A) divide and conquer (B) dynamic programming (C) greedy (D) heuristic (E) approximation

(44) The letter e should be coded as
(A) 01 (B) 1000 (C) 101 (D) 11 (E) 0

(45) The string 01100001011 should be decoded as
(A) bcda (B) abcde (C) aedfa (D) cdef (E) None of above

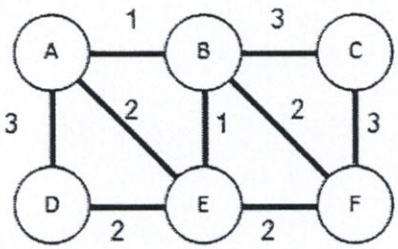



Figure 16.

**16.** This question is about the shortest path problem.

(46) Which of the following statements is wrong?
(A) Topological sort can be used to find the shortest paths when the input graph is a DAG.
(B) The Bellman-Ford algorithm can find the shortest path from a single source.
(C) The Floyd-Warshall algorithm can find all pairs of shortest paths.
(D) When the input graph is a directed acyclic graph, then the shortest path can be found in O(V+E) time.
(E) The Dijkstra algorithm cannot find the shortest path correctly when there is a cycle.

(47) Consider an undirected graph G(V, E) as in the Figure 16. Let vertex A be the source, then which of the following statements is correct?
(A) The edge DE can appear in a shortest path tree.
(B) The edge CF can appear in a shortest path tree.
(C) The edge EF can appear in a shortest path tree.
(D) The edge BF can appear in a shortest path tree.
(E) The shortest path tree has 6 edges.

(48) Consider the following procedure, which of the following statements is wrong?

```
RELAX(u, v, w) {
1.  If v.d > u.d + w(u, v)      // v.d is the current shortest distance from source to v
2.      v.d = u.d + w(u, v)
3.      v.p = u }
```

(A) This procedure is used in the Dijkstra algorithm.
(B) It takes O(1) amortized time for each call of RELAX by implementing the Dijkstra algorithm with a Fibonacci heap.
(C) It takes O(log V) amortized time for each call of RELAX by implementing the Dijkstra algorithm with a binary heap.
(D) This procedure is used in the Bellman-Ford algorithm as well.
(E) Implementing the Dijkstra algorithm with a priority-queue, Step 2 relates to the decrease-key operation.

**17.** This problem is about the minimum spanning tree problem.
(49) Consider the graph as shown in Figure 16. Which edge does not appear in a minimum spanning tree?
(A) BE, (B) AE, (C) CF, (D) BC, (E) DE
(50) Consider the following algorithm with the disjoint set data structure to find a minimum spanning tree.

```
MST(G, w)
1. A = empty set
2. For each vertex v Make-Set(v)
3. Sort the edges of G into non-decreasing order by weight
4. For each edge (u, v) in G taken in non-decreasing order by weight
5.     If _____
6.         add (u, v) to A
7.         Union(u, v)
8. Return A
```

What is the missing statement in line 5?
(A) u=v
(B) Find-Set(u) = Find-Set(v)
(C) Find-Set(u) == Find-Set(v)
(D) Find-Set(u) != Find-Set(v)
(E) u!=v

(51) What is the weight of the minimum spanning tree of Figure 16?
(A)7　(B) 8　(C) 9　(D) 10　(E) 11

**18.** This question is about the maximum flow problem.
(52) Suppose the capacity of each edge is integral. Which of the following statements is wrong?
(A) It is possible to have integral flows on each edge.
(B) It is possible to have fractional flow on some edge.
(C) The Push-Relabel algorithm can find the maximum flow in polynomial time.
(D) The Edmonds-Karp algorithm can find the maximum flow in $O(V^2E)$ time
(E) The Edmonds-Karp algorithm can find the augmenting path with DFS.

(53)Which statement is wrong?
(A) The flow across any cut is at most the capacity of the cut.
(B) The maximum flow is equal to the capacity of a minimum cut.
(C) The corresponding residual network of a maximum flow has no augmenting path between any 2 nodes.
(D) The length of augmenting path between the source and sink is nondecreasing in the Edmonds-Karp algorithm.
(E) The Ford-Fulkerson algorithm does not have a specific way to find the augmenting path.

(54) Which of the following is correct?
(A) If there is no augmenting path in a residual network, then the corresponding flow is maximum.
(B) If all edges of a flow network have different capacities, then th maximum flow is unique.
(C) The maximum bipartite matching can be transformed into a maximum flow problem by assigning distinct capacities to different edges.
(D) The residual capacity of an edge can be negative.
(E) For each edge, its flow can be larger than its capacity.

**19.** Consider the following graph algorithm.

```
A(G, w)      // w defines the weights of the edges
1. Compute G', where G' is obtained from G by adding a new vertex s and new edges (s,
   v) for all v in G(V), and with w(s, v)=0
2. If Bellman-Ford(G', w, s) == False    // s is the starting vertex
       Print "the input has a negative weight cycle"
```

```
3. Else for each vertex v in G'
        Set h(v) to be the shortest distance from s to v, computed by the above
        Bellman-Ford algorithm
4.      for each edge (u, v) of G' set w'(u, v) = w(u,v) + h(u) – h(v)
5.      let D = (d(u, v)) be a new n by n matrix
6.      for each vertex u in G(V)
7.          run Dijkstra's algorithm on G' with w' and u as the starting vertex
8.              to compute the shortest path distance dist(u, v) from u to v
                for all v in G(V).
9.          for each vertex v in G(V) set d(u, v) = dist(u, v) + h(v) – h(u)
10.     Return D
```

(55) Which of the following is wrong?
(A) For each edge (u, v) of G', w'(u, v) is non-negative.
(B) This algorithm finds all pairs of shortest paths of G.
(C) This algorithm is efficient when the number of edges is small.
(D) If there is no negative cycle in G', then for some edge (u, v), $h(v) > h(u) + w(u, v)$.
(E) h(v) can be negative.

(56) By using the Fibonacci heap, what is the complexity of the above algorithm?
(A) O(V+E) (B) $O(V^2 \lg V + VE)$ (C) $O(V^2)$ (D) O(E) (E) O(V lg V)

(57) Which of the following is wrong? Let $p = \langle u=v_0, v_1, \ldots, v_k=v \rangle$ be any path from vertex u to vertex v.
(A) p is a shortest path between u and v using weight function w if and only if p is also a shortest path using w'.
(B) w'(p) = w(p) + h(u) – h(v)
(C) The Dijkstra's algorithm is called O(V) times.
(D) If there is no cycle in G', then we can replace Bellman-Ford algorithm with the Dijkstra's algorithm
(E) It takes O(V) time to compute G' from G.

**20.** This question is about the NP-complete problems.
(58) Which of the following is wrong?
(A) If SAT has a polynomial time algorithm, then P =NP.
(B) SAT can be reduced to the Vertex Cover problem in polynomial time.
(C) 3-SAT problem is as difficult as SAT.
(D) 2-SAT can be reduced to 3-SAT and hence is NP-complete.
(E) SAT is NP-complete.

(59) Which of the following is correct?
(A) If a problem is NP-hard, then it must be NP-complete.
(B) Each problem in NP has a polynomial time verifying algorithm without any certificate.
(C) Any problem in NP-hard can be reduced to any NP-complete problem in polynomial time.
(D) If P = NP, then we can factorize any composite integer in polynomial time.
(E) Any problem that can be reduced to SAT is NP-complete.

(60) Which of the following is wrong?
(A) Any problem in P can be reduced to SAT in polynomial time.
(B) The problem to determine if a graph has a Hamiltonian path is in NP.
(C) It takes at most O(N!) time to determine if a graph of N vertices has a Hamiltonian path
(D) It takes at most $O(2^N)$ time to determine if a Boolean formula of N variables is satisfiable.
(E) A clause of a CNF boolean formula of length > 3 can be transformed into several clauses, where each clause has exactly 3 literals, which all come from the original clauses.