Test Analysis and Doubt Clearing Session

Rank Improvement Course on Algorithms

Subbarao Lingamgunta • Lesson 11 • Oct 31, 2021

If you like this class, follow my profile at
https://unacademy.com/@reddysir
to get all my courses/classes notifications

Don't forget to dedicate HAT'S

Time complexity to find
Out-degree of each vertex of the given directed tree which is represented by adjacency list representation?

A. O(V^2)
B. O(V+E)
C. O(V)
D. O(E)

Asha and Lata play a game in which Lata first thinks of a natural number between 1 and 1000. Asha must find out that number by asking Lata questions, but Lata can only reply by saying “Yes” or “no”. Assume that Lata always tells the truth. What is the least number of questions that Asha needs to ask within which she can always find out the number Lata has thought of?

A. 10
B. 42
C. 100
D. 1000

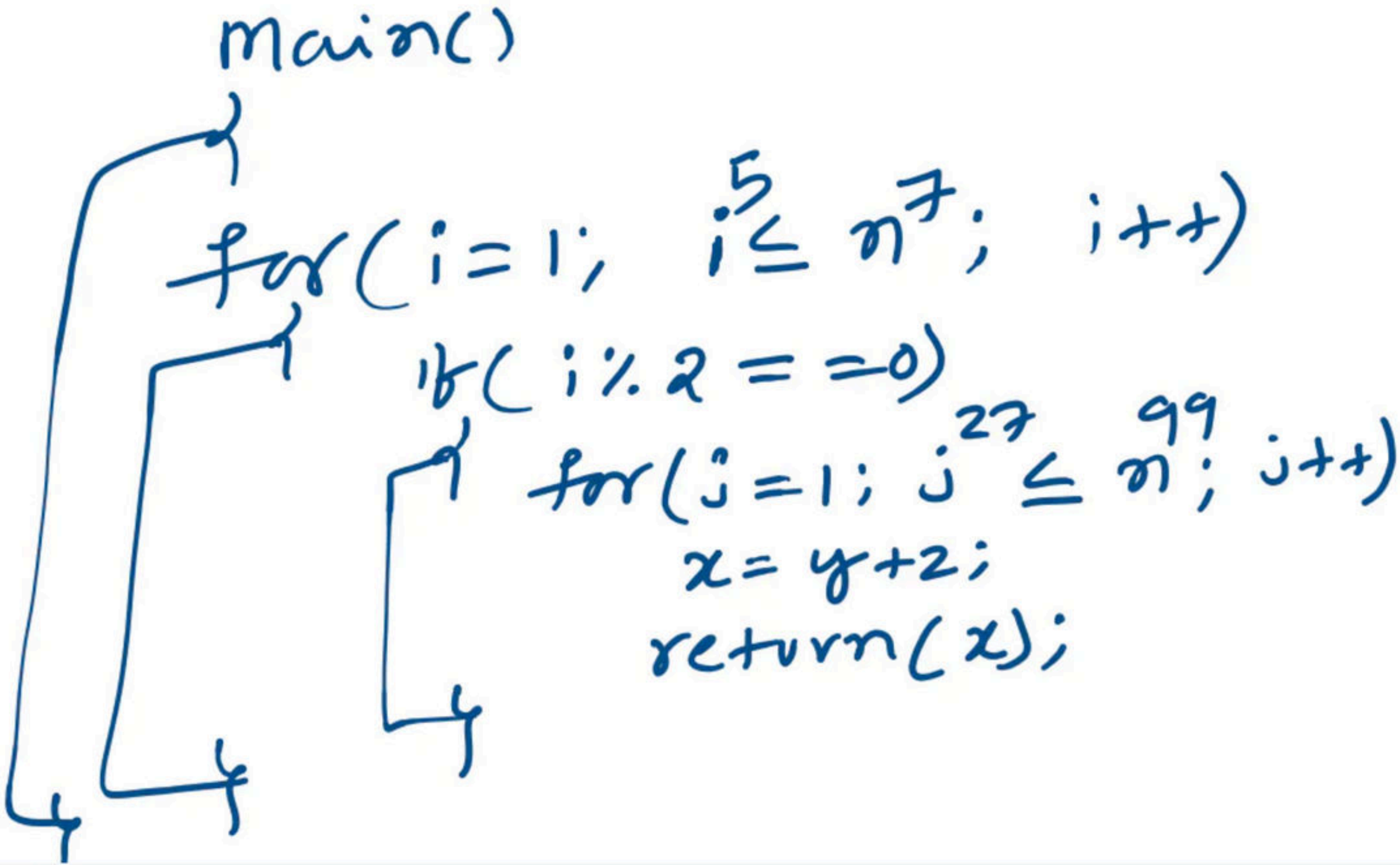
The minimum no. of. comparisions needed to determine if an integer appears more than $n/2$ times in sorted array of n-elements (or) not ?

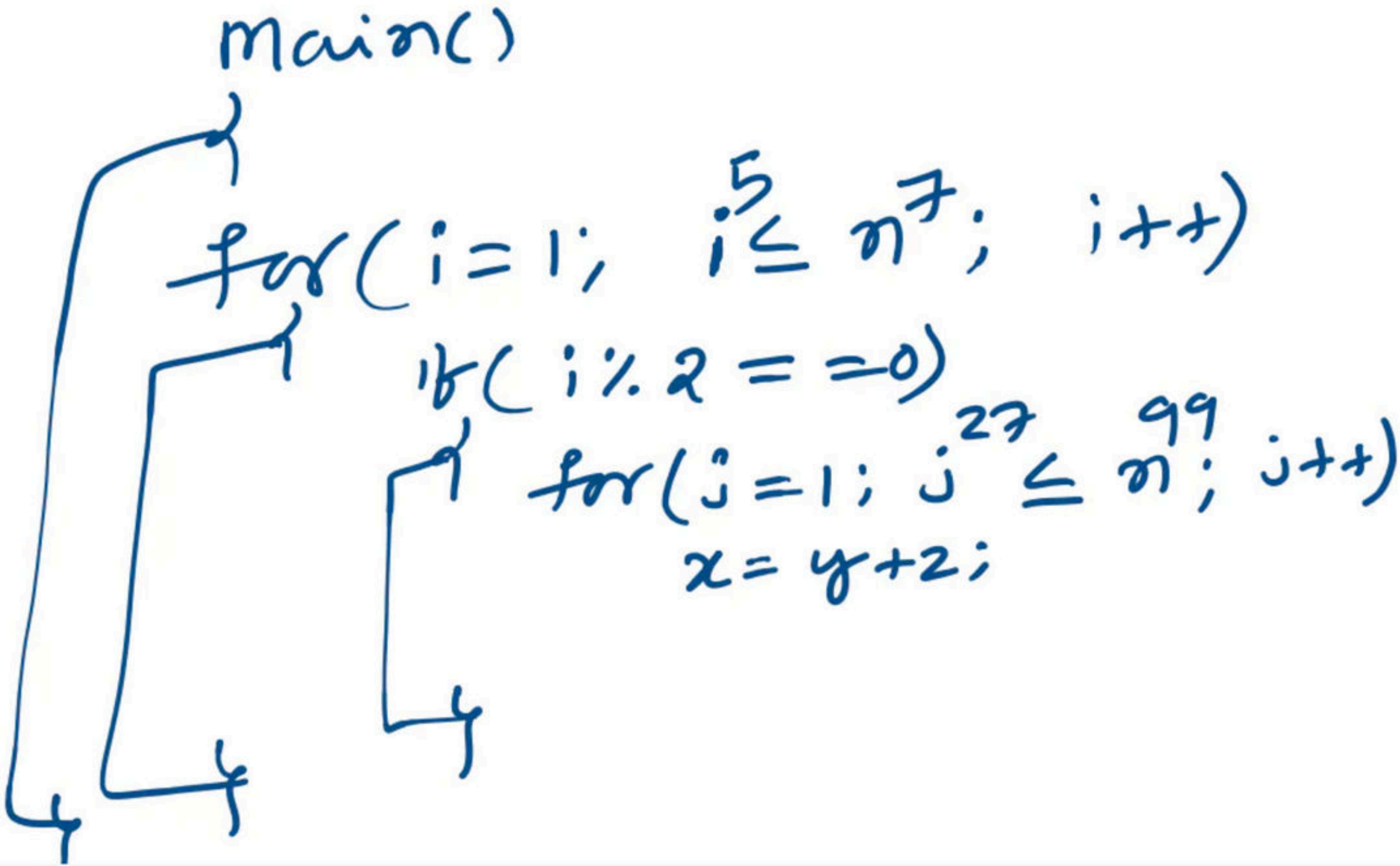
a) $\theta(n)$ b) $\theta(\log n)$ c) $\theta(1)$ d) $\theta(n^2)$

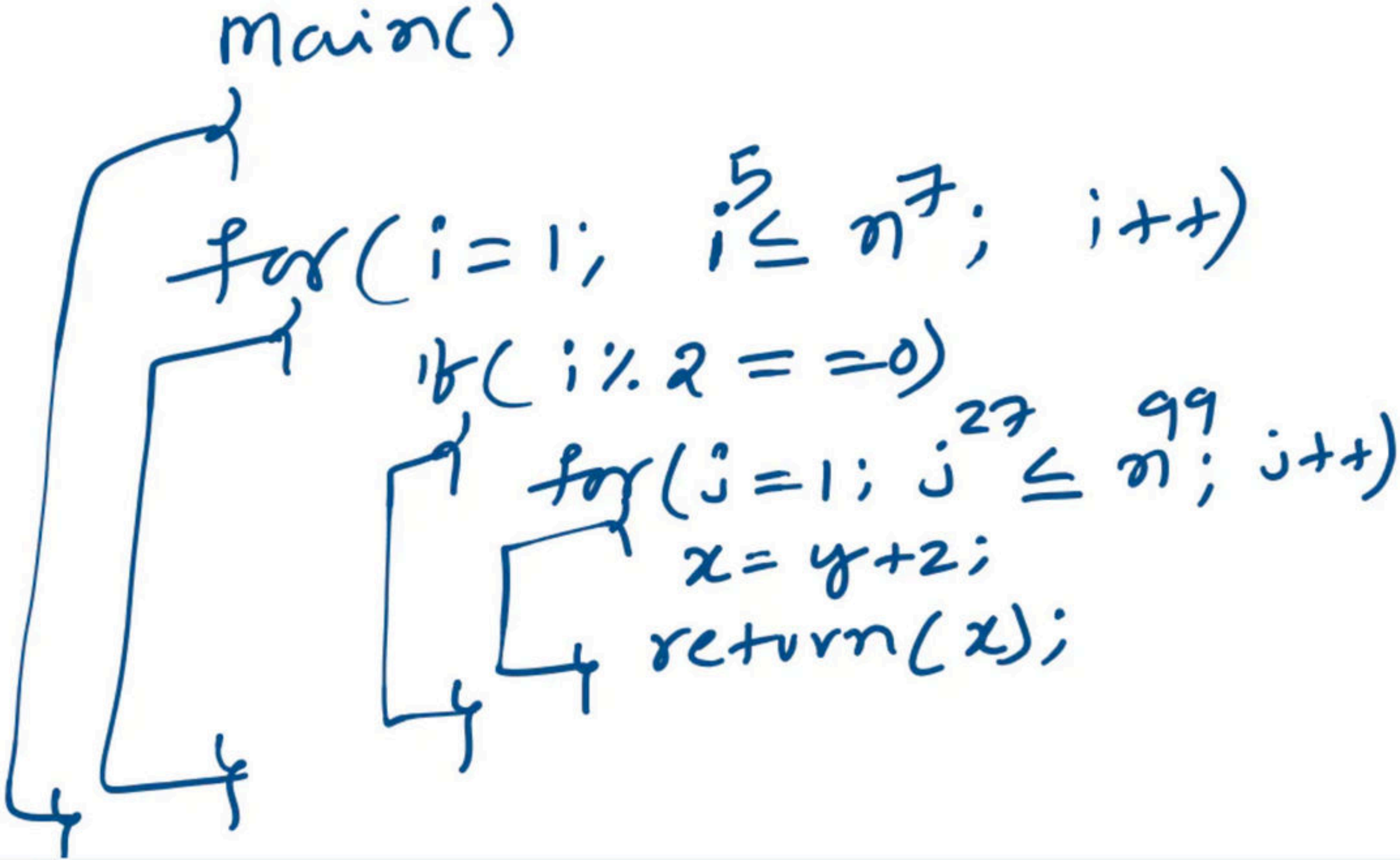
It is confirmed that in the given Sorted array of n-distinct elements one ele repeating more than n/2 times, what is the worst case TC to find that element?

The minimum no. of. comparisions needed to determine if an integer appears more than $n/2$ times in an array of n-elements

a) $\theta(n)$ b) $\theta(\log n)$ c) $\theta(1)$ d) $\theta(n^2)$


main()
for(i=1; i<=n^5; i++)
if(i%2==0)
for(j=1; j^27<=n^99; j++)
x=y+2;
return(x);


main()
for(i=1; i<=n; i++)
if(i%2==0)
for(j=1; j<=n; j++)
x=y+2;


main()
for(i=1; i<=n^5; i++)
if(i%2==0)
for(j=1; j^27<=n^99; j++)
x=y+2;
return(x);

```
unknown (int n)
{
  int i, j, k = 0;
  for (i = n/2; i <= n; i++)
    for (j = 2; j <= n; j = j*2)
      k = k + n/2;
  return (k);
}
```

The return value of the function is

(a) $\Theta(n^2)$ (b) $\Theta(n^2 \log n)$

(c) $\Theta(n^3)$ (d) $\Theta(n^3 \log n)$

```
unknown (int n)
{
  int i, j, k = 1;
  for (i = n/2; i <= n; i++)
    for (j = 2; j <= n; j = j*2)
      k = k * n/2;
  return (k);
}
```

The return value of the function is

(a) $\Theta(n^2)$ (b) $\Theta(n^2 \log n)$

(c) $\Theta(n^3)$ (d) $\Theta(n^3 \log n)$

```
unknown (int n)
{
  int i, j, k = 1;
  for (i = n/2; i <= n; i = 2*i)
    for (j = 2; j <= n; j = j*2)
      k = k * n/2;
  return (k);
}
```

The return value of the function is

(a) $\Theta(n^2)$

(b) $\Theta(n^2 \log n)$

(c) $\Theta(n^3)$

(d) $\Theta(n^3 \log n)$

```
int fun1 (int n)
{       int i, j, k, p, q = 0;
        for (i = 1; i < n; ++i)
        {
                for (j = n; j > 1; j = j/2)
                ++p;
                for (k = 1; k < p; k = k*2)
                ++q;
        } return q;
}
```

```
int fun1 (int n)
{       int i, j, k, p, q = 0;
        for (i = 1; i < n; ++i)
        {       p=0, q=0;
                for (j = n; j > 1; j = j/2)
                ++p;
                for (k = 1; k < p; k = k*2)
                ++q;
        } return q;
}
```

```
int fun1 (int n)
{     int i, j, k, p, q = 0;
      for (i = 1; i < n; ++i)
      {     p=0;
            for (j = n; j > 1; j = j/2)
            ++p;
            for (k = 1; k < p; k = k*2)
            ++q;
      } return q;
}
```

Find Time complexity of q-value of above 3-algo's.

An algorithm performs $(\log N)^{1/2}$ find operations, N insert operations, $(\log N)^{1/2}$ delete operations, and $(\log N)^{1/2}$ decrease-key operations on a set of data items with keys drawn from a linearly ordered set. For a delete operation, a pointer is provided to the record that must be deleted. For the decrease-key operation, a pointer is provided to the record that has its key decreased. Which one of the following data structures is the most suited for the algorithm to use, if the goal is to achieve the best total asymptotic complexity considering all the operations?

(A) Unsorted array

(B) Min-heap

(C) Sorted array

(D) Sorted doubly linked list

```
void Function(int n) {
        int i=1, s=1;
        while( s <= n) {
                i++;
                s= s+i;
                printf("*");
        }
}
```

TC ?

```
void Function(int n) {
        int i=1, s=1;
        while( s <= n) {
                i++;
                s= s+i;
                printf("*");
        }
}
```

Tc ?

There are n unsorted arrays: A_1, A_2, ….,A_n. Assume that n is odd. Each of A_1, A_2, …., A_n contains n distinct elements. There are no common elements between any two arrays. The worst-case time complexity of computing the median of the medians of A_1, A_2, ….,A_n is _____ .

(A) O(n log n)

(B) $O(n^2)$

(C) O(n)

(D) $\Omega(n^2 \log n)$

Let G=(V,E) be a weighted undirected graph and let T be a Minimum Spanning Tree (MST) of G maintained using adjacency lists. Suppose a new weighed edge $(u, v) \in V \times V$ is added to G. The worst-case time complexity of determining if T is still an MST of the resultant graph is

A. $\Theta(|E|+|V|)$
B. $\Theta(|E||V|)$
C. $\Theta(E|\log|V|)$
D. $\Theta(|V|)$

Let G=(V,E) be a weighted undirected graph and let T be a Minimum Spanning Tree (MST) of G maintained using adjacency Matrix. Suppose a new weighed edge $(u, v) \in V \times V$ is added to G. The worst-case time complexity of determining if T is still an MST of the resultant graph is

A. $\Theta(|E|+|V|)$
B. $\Theta(|E||V|)$
C. $\Theta(E|\log|V|)$
D. $\Theta(|V|)$

Let A[1, …, n] be an array storing a bit (1 or 0) at each location, and f(m) is a function whose time complexity is θ(m). Consider the following program fragment written in a C like language:

```
counter = 0;
for (i = 1; i < = n; i++)
{
        if (A[i] == 1)
                counter++;
        else {
                f(counter);
                counter = 0;
        }
}
```

The complexity of this program fragment is

(A) $\Omega(n^2)$

(B) $\Omega(n \log n)$ and $O(n^2)$

(C) $\theta(n)$

(D) $O(n)$

Let A[1, …, n] be an array storing a bit (1 or 0) at each location, and f(m) is a function whose time complexity is θ(m). Consider the following program fragment written in a C like language:

```
counter = 0;
for (i = 1; i < = n; i++)
{
        if (A[i] == 1)
                counter++;
        else {
                f(counter);
              }
}
```

The complexity of this program fragment is

(A) $\Omega(n^2)$

(B) $\Omega(n \log n)$ and $O(n^2)$

(C) $\theta(n)$

(D) $O(n)$

Let A[1, …, n] be an array storing a bit (1 or 0) at each location, and f(m) is a function whose time complexity is θ(n). Consider the following program fragment written in a C like language:

```
counter = 0;
for (i = 1; i < = n; i++)
{
        if (A[i] == 1)
                counter++;
        else {
                f(counter);
                counter = 0;
        }
}
```

The complexity of this program fragment is

(A) $\Omega(n^2)$

(B) $\Omega(n \log n)$ and $O(n^2)$

(C) $\theta(n)$

(D) $O(n)$

```
int f1(int n)
{
if(n == 0 || n == 1)
        return n;
else
        return (2*f1(n-1) + 3*f1(n-2));
}

int f2(int n)
{
int i;
int X[N], Y[N], Z[N] ;
X[0] = Y[0] = Z[0] = 0;
X[1] = 1; Y[1] = 2; Z[1] = 3;
for(i = 2; i <= n; i++)
{
        X[i] = Y[i-1] + Z[i-2];
        Y[i] = 2*X[i];
        Z[i] = 3*X[i];
}
return X[n] ;
}
```

TC of f_1 & f_2

(a) $n, 2^n$

(b) $2^n, n$

(c) $2^n, 2^n$

(d) n, n

```
double foo (int n)
{
        int i;
        double sum;
        if (n = = 0) return 1.0;
        else
        {
                sum = 0.0;
                for (i = 0; i < n; i++)
                        sum += foo (i);
                return sum;
        }
}
```

find space complexity ?

Suppose we modify the above function foo() and store the values of foo (i), 0 < = i < n, as and when they are computed. With this modification, the time complexity for function foo() is significantly reduced.

The space complexity of the modified function would be:

(A) O(1)

(B) O(n)

(C) O(n!)

(D) $O(n^n)$

In the given array of n-distinct elements to find any element which is neither minimum nor maximum will take how much time?

In the given array of n-elements to find any element which is either minimum nor maximum how much time?

In the given **sorted** array of n-distinct elements to find **any** element which is neither minimum nor maximum will take how much time?

In the given sorted array of n-elements to find any element which is neither minimum nor maximum how much time?

In the given array of n-distinct elements to find any element which is neither 10^{th} minimum nor 10^{th} maximum how much time it will take?

In the given array of n-distinct elements to find any element which is neither K^{th} maximum nor K^{th} minimum how much time?

How much time it will take to find universal sink in the given directed graph of n-vertices represented in adjacency matrix.

Note: Universal Sink ⟺ Outdegree — 0
Indegree — V-1

TC?

How much time it will take to find universal sink in the given directed graph of n-vertices represented in Adjacency List.

Note: Universal Sink

TC?

Outdegree — 0
Indegree — V-1

Universal sink in a directed graph is a vertex i such that there is an edge from every vertex j (j≠ i) to i and there is no edge from i to any other vertex. A directed graph G with n vertices is represented by its adjacency matrix A, where A[i] [j] = 1 if there is an edge directed from each vertex i to j and 0 otherwise. The following algorithm determines whether there is a universal sink in the graph G.

```
i = 0
do
{
j = i + 1;
while ((j < n) && E1)
 j++;
if (j < n)
E2;
 }  while (j < n);

flag = 1;
 for (j = 0; j < n; j++)
if ((j! = i) && E3) flag = 0;

if (flag) printf("Universal Sink exists");
else printf ("Universal Sink does not exist");
```

1. Choose the correct expressions for E_1 and E_2

(A) E1 : A[i][j] and E2 : i = j;

(B) E1 : !A[i][j] and E2 : i = j + 1;

(C) E1: !A[i][j] and E2 : i = j;

(D) E1 : A[i][j] and E2 : i = j + 1;

2. Choose the correct expression for E3

A. (A[i][j] && !A[j][i])
B. (!A[i][j] && A[j][i])
C. (!A[i][j] ||A[j][i])
D. (A[i][j] || !A[j][i])

```
A(n)
{
    if(n <= 1)  return(n^10);
    else
    return(A(√n) + n + logn + n^3);
}
```

TC ?

```
A(n)
{
    if(n<=1) return(n^10);
    else
    return(A(n/2) * A(n/2) + A(n/2));
}
```

TC ?

```
A(n)
{
  if(n<=1) return(n!0);
  else
  return(n^2*A(n/2)-n(A(n/2)));
}
```

TC ?

Consider the following program:

```
1.int Bar(int n)
2.{
3.if(n<2) return;
4.else{
5.int sum=0;
6.int i,j;
7.for(i=1;i<=4;i++)
8.Bar(n/2);
9.for(i=1;i<=n;i++)
10.for(j=1;j<=i;j++)
11.sum=sum+1;
12.}
13.}
```

Now consider the following statement

$S1$: The time complexity of $Bar(n)$ is $\Theta(n2\text{^}logn)$
$S2$:The time complexity of $Bar(n)$ is $\Omega(n2\text{^}logn2)$
$S3$:The time complexity of $Bar(n)$ is $O(n3\text{^}logn2)$

How many statements are correct----------

```
void function(int n) {
        int i, j, k , count =0;
        for(i=n/2; i<=n; i++)
                for(j=1; j + n/2<=n; j= j++)
                        for(k=1; k<=n; k= k * 2)
                                count++;
}
```

Consider the following algorithm for searching for a given number x in an unsorted array A[1.....n] having n distinct values:

1. Choose an i uniformly at random from 1..... n;

2. If A[i] = x then Stop else Go to 1;

Assuming that x is present in A, what is the maximum number of comparisons made by the algorithm before it terminates ?

(A) n

(B) n – 1

(C) 2n

(D) n/2

Consider the following algorithm for searching for a given number x in an unsorted array A[1.....n] having n distinct values:

1. Choose an i uniformly at random from 1..... n;

2. If A[i] = x then Stop else Go to 1;

What is the maximum number of comparisons made by the algorithm before it terminates?

(A) n

(B) n – 1

(C) 2n

(D) n/2

Consider the following algorithm for searching for a given number x in an unsorted array A[1.....n] having n distinct values:

1. **Choose an i uniformly at random from 1..... n;**
2. **If A[i] = x then Stop else Go to 1;**

Assuming that x is present in A, what is the expected number of comparisons made by the algorithm before it terminates ?

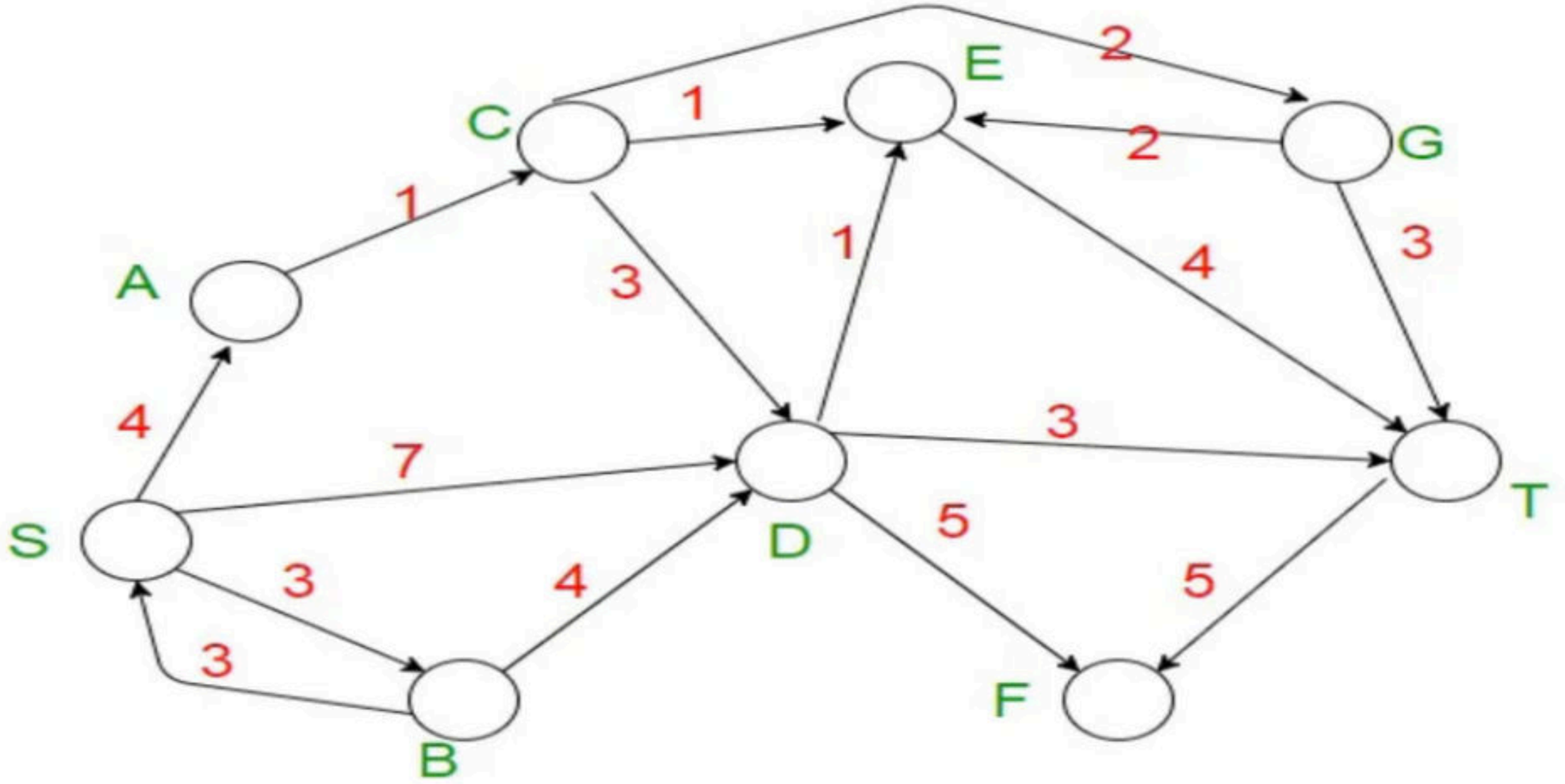
(A) n

(B) n – 1

(C) 2n

(D) n/2

Consider the directed graph shown in the figure below.



Assume that, in any iteration, the shortest path to a vertex v is updated only when a strictly shorter path to v is discovered. Staring from vertex s what will will be the sequence of vertices identified by Dijkstra's algorithms?

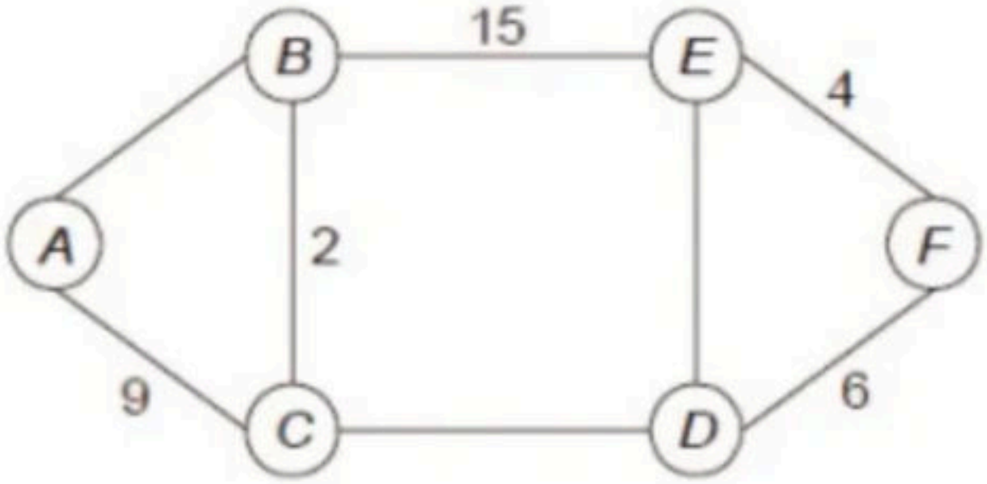
(A) SABCDEFGT
(B) SBDTGACEF
(C) SACDTBGEF
(D) SACETDBFG

There are multiple shortest paths between vertices S and T. Which one will be reported by Dijkstra's shortest path algorithm? Assume that, in any iteration, the shortest path to a vertex v is updated only when a strictly shorter path to v is discovered.

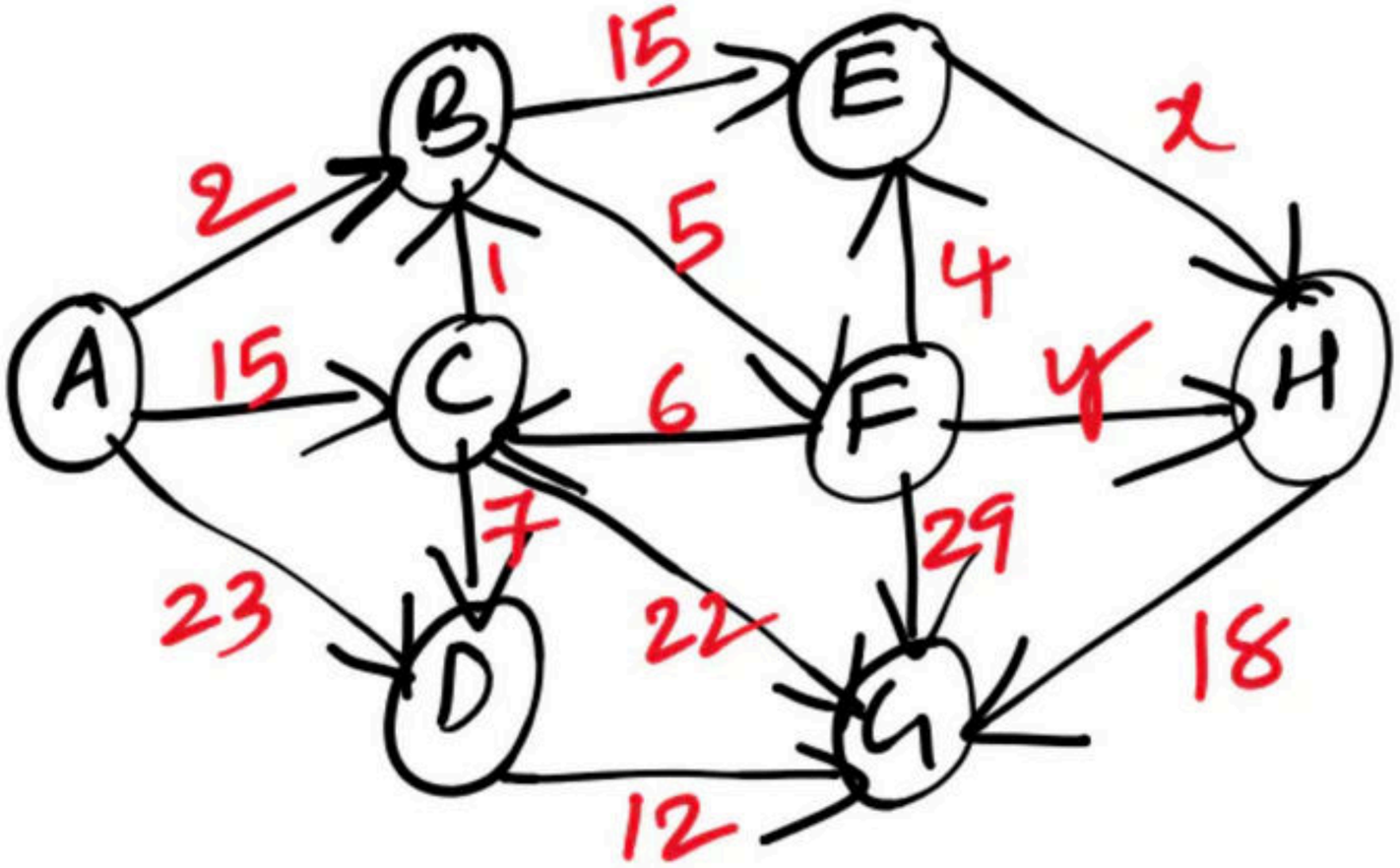
(A) SDT
(B) SBDT
(C) SACDT
(D) SACET

The graph shown below 8 edges with distinct integer edge weights. The minimum spanning tree (MST) is of weight 36 and contains the edges: {(A, C), (B, C), (B, E), (E, F), (D, F)}. The edge weights of only those edges which are in the MST are given in the figure shown below. The minimum possible sum of weights of all 8 edges of this graph is---------

(A) 66
(B) 69
(C) 68
(D) 70



Suppose that you are running Dijkstra's algorithm on the edge-weighted diagram below, starting from vertex A.



The Table gives 'Distance' and 'Parent' entry of each vertex after vertex E has been deleted from the priority queue and relaxed.

Vertex	Distance	Parent
A	0	null
B	2	A
C	13	F
D	23	A
E	11	F
F	7	B
G	36	F
H	19	E

What could be the possible value of expression x+y?

Consider the following statements

I. Let T be a minimum spanning tree of a graph G. Then for any two vertices u and v the path from u to v in T is the shortest path from u to v in the graph G

II. Suppose that average edge weight for a graph G is AG, Then the minimum spanning tree of G will have weight at most (n-1)AG, where n is the number of vertices in graph G.

which of the above statements are true ?

A)Only I B)Only II C)both I and II D)None of these

Given two arrays of numbers a1, a2, a3,...an and b1, b2, .. bn where each number is 0 or 1, the fastest algorithm to find the largest span(i, j) such that ai ai+1, ...aj = bi , bi+1, .. bj . or report that there is not such span.

(a) Takes O (n^3) and Ω (2^n) time if hashing is permitted
(b) Takes O (n^3) and Ω (n^2.5) time in the key comparison model
(c) Takes θ (n) time and space
(d) Takes O (√n) time only if the sum of the 2n elements is an even number

The weight of a sequence $a_0, a_1, \ldots, a_{n-1}$ of real numbers is defined as $a_0+a_1/2+\ldots+ a_{a-1}/2^{n-1}$. A subsequence of a sequence is obtained by deleting some elements from the sequence, keeping the order of the remaining elements the same. Let X denote the maximum possible weight of a subsequence of $a_0, a_1, \ldots,a_{n-1}$ and Y the maximum possible weight of a subsequence of $a_1, a_2, \ldots,a_{n-1}$. Then X is equal to

(A) max(Y, a0+Y)

(B) max(Y, a0+Y/2)

(C) max(Y, a0+2Y)

(D) a0+Y/2

An articulation point in a connected graph is a vertex such that removing the vertex and its incident edges disconnects the graph into two or more connected components.
Let T be a DFS tree obtained by doing DFS in a connected undirected graph G. Which of the following options is/are correct?

A. Root of T can never be an articulation point in G.

B. Root of T is an articulation point in G if and only if it has 2 or more children.

C. A leaf of T can be an articulation point in G.

D. If u is an articulation point in G such that x is an ancestor of u in T and y is a descendent of u in T, then all paths from x to y in G must pass through u.

G is a graph on n vertices and 2n – 2 edges. The edges of G can be partitioned into two edge-disjoint spanning trees. Which of the following is NOT true for G?

(A) For every subset of k vertices, the induced subgraph has at most 2k-2 edges
(B) The minimum cut in G has at least two edges
(C) There are two edge-disjoint paths between every pair to vertices
(D) There are two vertex-disjoint paths between every pair of vertices

In an adjacency list representation of an undirected simple graph G = (V, E), each edge (u, v) has two adjacency list entries: [v] in the adjacency list of u, and [u] in the adjacency list of v. These are called twins of each other. A twin pointer is a pointer from an adjacency list entry to its twin. If |E| = m and |V | = n, and the memory size is not a constraint,
what is the time complexity of the most efficient algorithm to set the twin pointer in each entry in each adjacency list?

(A) $\Theta(n^2)$
(B) $\Theta(m+n)$
(C) $\Theta(m^2)$
(D) $\Theta(n^4)$

Consider the weighted undirected graph with 4 vertices, where the weight of edge {i, j} g is given by the entry W_{ij} in the matrix W

The largest possible integer value of x, for which at least one shortest path between some pair of vertices will contain the edge with weight x is------

(A) 8
(B) 12
(C) 10
(D) 11

$$W = \begin{bmatrix} 0 & 2 & 8 & 5 \\ 2 & 0 & 5 & 8 \\ 8 & 5 & 0 & x \\ 5 & 8 & x & 0 \end{bmatrix}$$

Suppose you want to move from 0 to 100 on the number line. In each step, you either move right by a unit distance or you take a shortcut. Suppose T(k) denotes the smallest number of steps needed to move from k to 100. Suppose further that there is at most 1 shortcut involving any number, and in particular from 9 there is a shortcut to 15. Let y and z be such that T(9) = 1 + min(T(y), T(z)). Then the value of the product yz is ------

(A) 50
(B) 100
(C) 150
(D) 200

A sub-sequence of a given sequence is just the given sequence with some elements (possibly none or all) left out. We are given two sequences X[m] and Y[n] of lengths m and n respectively, with indexes of X and Y starting from 0. We wish to find the length of the longest common sub-sequence(LCS) of X[m] and Y[n] as l(m,n), where an incomplete recursive definition for the function l(i,j) to compute the length of The LCS of X[m] and Y[n] is given below:

l(i,j) = 0, if either i=0 or j=0
= expr1, if i,j > 0 and X[i-1] = Y[j-1] ;
= expr2, if i,j > 0 and X[i-1] != Y[j-1];

(A) expr1 ≡ l(i-1, j) + 1
(B) expr1 ≡ l(i, j-1)
(C) expr2 ≡ max(l(i-1, j), l(i, j-1))
(D) expr2 ≡ max(l(i-1,j-1),l(i,j))

Consider the data given in the above, the values of l(i, j) could be obtained by dynamic programming based on the correct recursive definition of l(i, j) of the form given above, using an array L[M, N], where M = m+1 and N =n+1, such that L[i, j] = l(i, j).
Which one of the following statements would be TRUE regarding the dynamic programming solution for the recursive definition of l(i, j)?

(A) All elements L should be initialized to 0 for the values of l(i,j) to be properly computed
(B) The values of l(i,j) may be computed in a row major order or column major order of L(M,N)
(C) The values of l(i,j) cannot be computed in either row major order or column major order of L(M,N)
(D) L[p,q] needs to be computed before L[r,s] if either p < r or q < s.

Assume that multiplying a matrix G_1 of dimension p×q with another matrix G_2 of dimension q×r requires pqr scalar multiplications. Computing the product of n matrices $G_1G_2G_3$ ….. G_n can be done by parenthesizing in different ways. Define G_iG_{i+1} as an explicitly computed pair for a given parameterisation if they are directly multiplied. For example, in the matrix multiplication chain $G_1G_2G_3G_4G_5G_6$ using parameterization $(G_1(G_2G_3))(G_4(G_5G_6))$, G_2G_3 and G_5G_6 are only explicitly computed pairs. Consider a matrix multiplication chain $F_1F_2F_3F_4F_5$, where matrices F_1,F_2,F_3,F_4 and F_5 are of dimensions 2×25,25×3,3×16,16×1 and 1×1000, respectively. In the parameterization of $F_1F_2F_3F_4F_5$ that minimizes the total number of scalar multiplications, the explicitly computed pairs is/are

(A) F_1F_2 and F_3F_4 only
(B) F_2F_3 only
(C) F_3F_4 only
(D) F_1F_2 and F_4F_5 only

Let A be a sequence of 8 distinct integers sorted in ascending order. How many distinct pairs of sequences, B and C are there such that (i) each is sorted in ascending order, (ii) B has 5 and C has 3 elements, and (iii) the result of merging B and C gives A?

(A) 2
(B) 30
(C) 56
(D) 256

Consider the process of inserting an element into a MaxHeap, where the MaxHeap is represented by an array. Suppose we perform a binary search on the path from the new leaf to the root to find the position for the newly inserted element, the number of comparisons performed is:

A. $\Theta(\log n)$
B. $\Theta(\log\log n)$
C. $\Theta(n)$
D. $\Theta(n\log n)$

The subset-sum problem is defined as follows. Given a set of n positive integers, S = {a1 ,a2 ,a3 ,...,an} and positive integer W, is there a subset of S whose elements sum to W? A dynamic program for solving this problem uses a 2-dimensional Boolean array X, with n rows and W+1 columns. X[i, j],1 <= i <= n, 0 <= j <= W, is TRUE if and only if there is a subset of {a1 ,a2 ,...,an} whose elements sum to j. Which of the following is valid for 1 <= i <= n and 0 <= j <= W?

(A) X[i, j] = X[i – 1, j] V X[i, j -ai]
(B) X[i, j] = X[i – 1, j] V X[i – 1, j – ai]
(C) X[i, j] = X[i – 1, j] X[i, j – ai]
(D) X[i, j] = X[i – 1, j] X[i -1, j – ai]

The subset-sum problem is defined as follows. Given a set of n positive integers, S = {a1 ,a2 ,a3 ,...,an} and positive integer W, is there a subset of S whose elements sum to W? A dynamic program for solving this problem uses a 2-dimensional Boolean array X, with n rows and W+1 columns. X[i, j],1 <= i <= n, 0 <= j <= W, is TRUE if and only if there is a subset of {a1 ,a2 ,...,an} whose elements sum to j.

Which entry of the array X, if TRUE, implies that there is a subset whose elements sum to W?

(A) X[1, W]
(B) X[n ,0]
(C) X[n, W]
(D) X[n -1, n]

Consider the following C program that attempts to locate an element x in an array Y[] using binary search. The program is erroneous.

```
1.  f(int Y[10], int x)
2.  {
2.          int i, j, k;
3.          i = 0; j = 9;
4.          do {
5.         k = (i + j) /2;
6.           if( Y[k] < x) i = k; else j = k;
7                  } while(Y[k] != x && i < j);
8.          if(Y[k] == x) printf ("x is in the array ") ;
9.          else printf (" x is not in the array ") ;
10. }
```

On which of the following contents of Y and x does the program fail?

(A) Y is [1 2 3 4 5 6 7 8 9 10] and x < 10

(B) Y is [1 3 5 7 9 11 13 15 17 19] and x < 1

(C) Y is [2 2 2 2 2 2 2 2 2 2] and x > 2

(D) Y is [2 4 6 8 10 12 14 16 18 20] and 2 < x < 20 and x is even

The correction needed in the program to make it work properly is

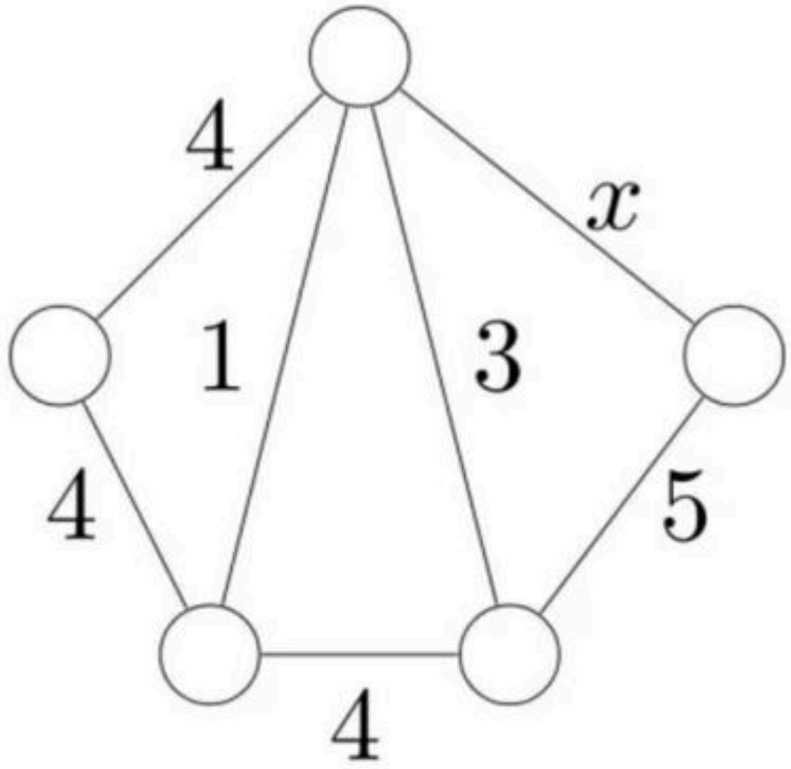
(A) Change line 6 to: if (Y[k] < x) i = k + 1; else j = k-1;

(B) Change line 6 to: if (Y[k] < x) i = k - 1; else j = k+1;

(C) Change line 6 to: if (Y[k] <= x) i = k; else j = k;

(D) Change line 7 to: } while ((Y[k] == x) && (i < j));

Choose a value for x that will maximize the number of minimum weight spanning trees (MWSTs) of G. The number of MWSTs of G for this value of x is-----



Consider the following functions from positives integers to real numbers

10, $\sqrt{n}$, n, $\log_2 n$, 100/n.

The CORRECT arrangement of the above functions in increasing order of asymptotic complexity is:

(A) $\log_2 n$, 100/n, 10, $\sqrt{n}$, n

(B) 100/n, 10, $\log_2 n$, $\sqrt{n}$, n

(C) 10, 100/n , $\sqrt{n}$, $\log_2 n$, n

(D) 100/n, $\log_2 n$, 10 , $\sqrt{n}$, n

An array A contains n integers. We wish to sort A in ascending order. We are told that initially no element of A is more than a distance k away from its final position in the sorted list. Assume that n and k are large and k is much smaller than n. Which of the following is true for the worst case complexity of sorting A?

a. A can be sorted with kn comparison but not with fewer comparisons.
b. A cannot be sorted with less than $n\log n$ comparisons.
c. A can be sorted with n comparisons.
d. A can be sorted with $n\log k$ comparisons but not with fewer comparisons.
e. A can be sorted with $k2n$ comparisons but not fewer.

Arrange the following functions in asymptotically increasing order

f1(n) = n^0.999999 logn. **f2(n) = 10000000n**

f3(n) = 1.000001^n **f4(n) = n^2**

(A) f1(n); f4(n); f2(n); f3(n)

(B) f1(n); f2(n); f3(n); f4(n)

(C) f2(n); f1(n); f4(n); f3(n)

(D) f1(n); f2(n); f4(n); f3(n)

Which of the given options provides the increasing order of asymptotic complexity of functions f1, f2, f3 and f4?

f1(n) = 2^n

f2(n) = n^(3/2)

f3(n) = nlogn

f4(n) = n^(logn)

(A) f3, f2, f4, f1

(B) f3, f2, f1, f4

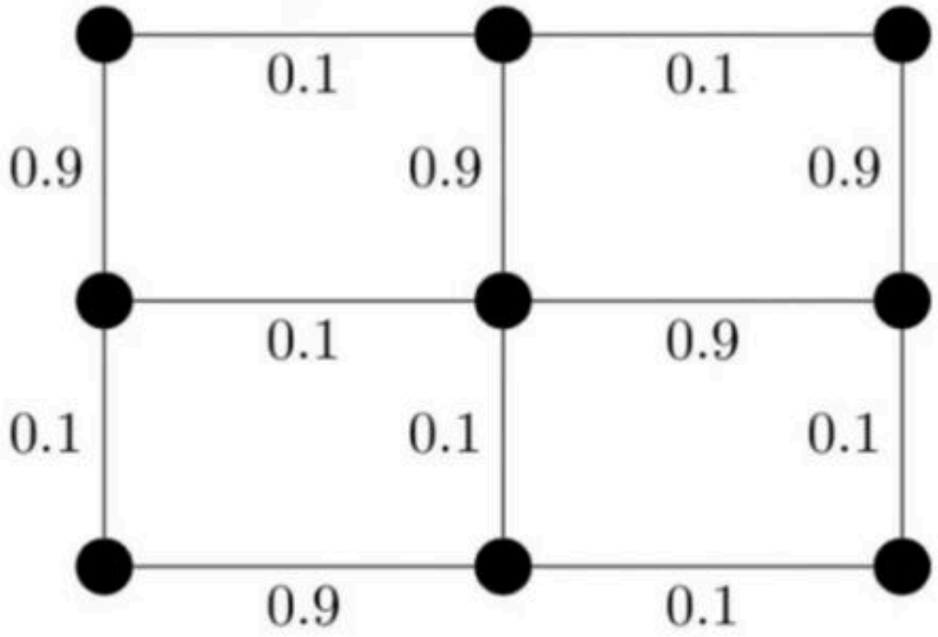
(C) f2, f3, f1, f4

(D) f2, f3, f4, f1

When $n = 2^{2k}$ for some $k \geq 0$, the recurrence relation
$T(n) = \sqrt{2}\, T(n/2) + \sqrt{n}$, $T(1) = 1$
evaluates to :

(A) $\sqrt{n}(\log n+1)$
(B) $\sqrt{n}(\log n)$
(C) $\sqrt{n}\log\sqrt{n}$
(D) $n\log\sqrt{n}$

Consider the following undirected graph with edge weights as shown:



The number of minimum-weight spanning trees of the graph is------

Let G be a simple undirected graph. Let TD be a depth first search tree of G. Let TB be a breadth first search tree of G. Consider the following statements.

1.No edge of G is a cross edge with respect to TD. (A cross edge in G is between two nodes neither of which is an ancestor of the other in TD).

2.For every edge (u,v) of G, if u is at depth i and v is at depth j in TB, then $|i-j|=1$.

Which of the statements above must necessarily be true?

a. I only

b. II only

c. Both I and II

d. Neither I nor II

In a permutation $a1...an...an$, of n distinct integers, an inversion is a pair (ai, aj) such that i<j and ai > aj. If all permutations are equally likely, what is the expected number of inversions in a randomly chosen permutation of 1.....n?

A. n(n−1)2
B. n(n−1)4
C. n(n+1)4
D. $2n[\log 2n]$

The tightest lower bound on the number of comparisons, in the worst case, for comparison-based sorting is of the order of

(A) N
(B) N^2
(C) NlogN
(D) N(logN)^2

A Young tableau is a 2D array of integers increasing from left to right and from top to bottom. Any unfilled entries are marked with ∞, and hence there cannot be any entry to the right of, or below a ∞.
The following Young tableau consists of unique entries.

1	2	5	14
3	4	6	23
10	12	18	25
31	∞	∞	∞

Then to search any element x in young tableau which contain n^2 elements will take how much time------

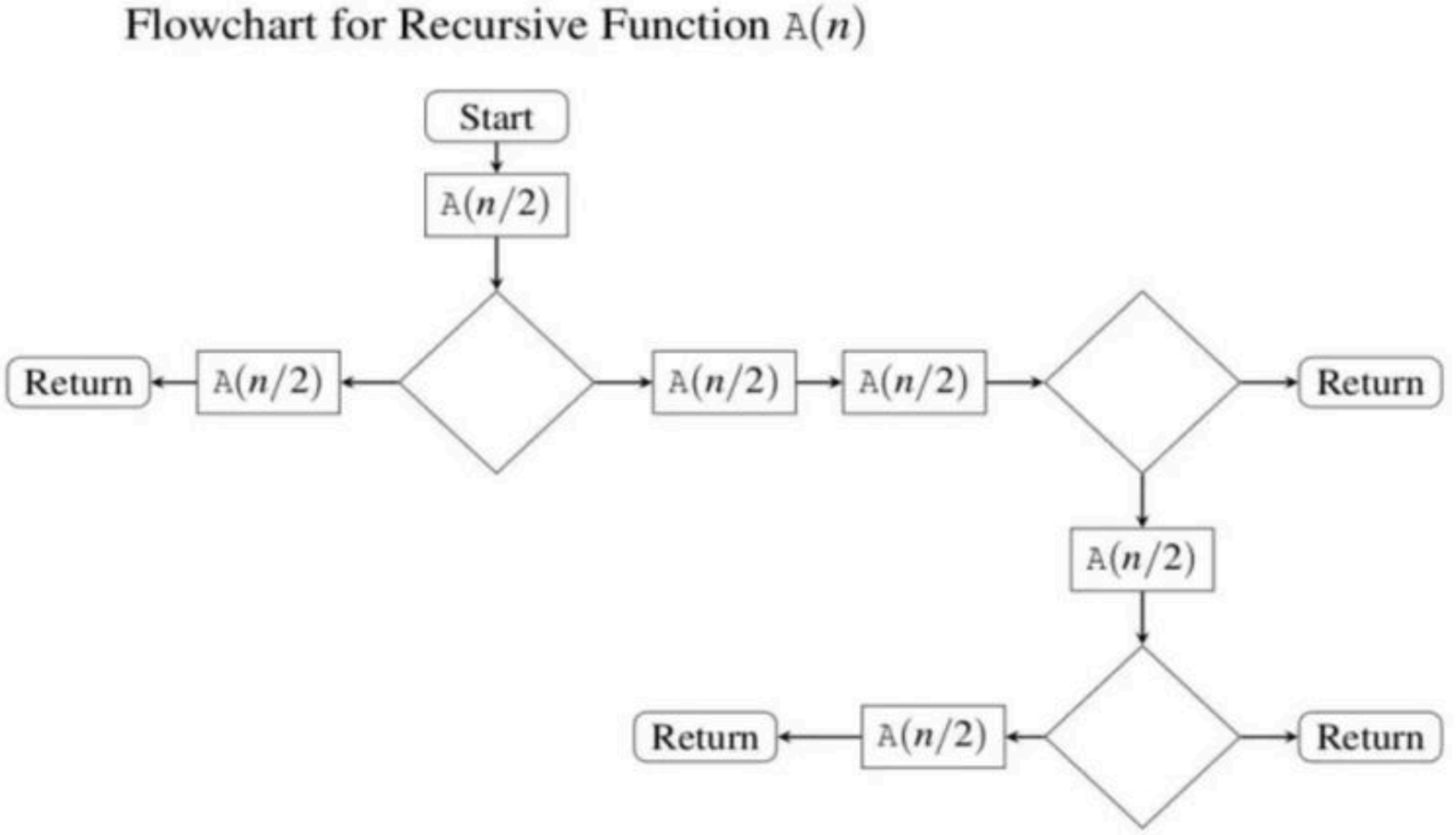
A Young tableau is a 2D array of integers increasing from left to right and from top to bottom. Any unfilled entries are marked with ∞, and hence there cannot be any entry to the right of, or below a ∞.
The following Young tableau consists of unique entries.

1	2	5	14
3	4	6	23
10	12	18	25
31	∞	∞	∞

then an element is removed from a Young tableau, other elements should be moved into its place so that the resulting table is still a Young tableau (unfilled entries may be filled in with a ∞).

The minimum number of entries (other than 1) to be shifted, to remove 1 from the given Young tableau is (A) 2 (B) 5 (C) 6 (D) 18

The given diagram shows the flowchart for a recursive function A(n). Assume that all statements, except for the recursive calls, have O(1) time complexity. If the worst case time complexity of this function is $O(n^{\alpha})$, then the least possible value (accurate up to two decimal positions) of α is --------



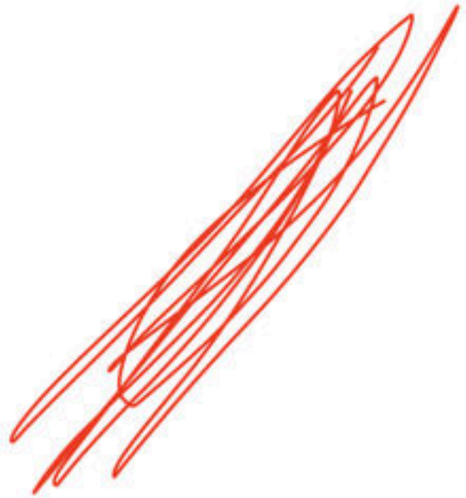
Let G = (V, E) be an undirected graph with a subgraph G1 = (V1, El). Weights are assigned to edges of G as follows

$$w(e) = \begin{cases} 0 & \text{if } e \in E_1 \\ 1 & \text{otherwise} \end{cases}$$

A single-source shortest path algorithm is executed on the weighted graph (V, E, w) with an arbitrary vertex v1 of V1 as the source. Which of the following can always be inferred from the path costs computed?

(A) The number of edges in the shortest paths from v1 to all vertices of G
(B) G1 is connected
(C) V1 forms a clique in G
(D) G1 is a tree

Suppose there are 4 sorted lists of n/4 elements each. if we merge these list into a single sorted list of n elements, for the n=400 number of key comparisons in the worst case using an efficient algorithm is



unacademy

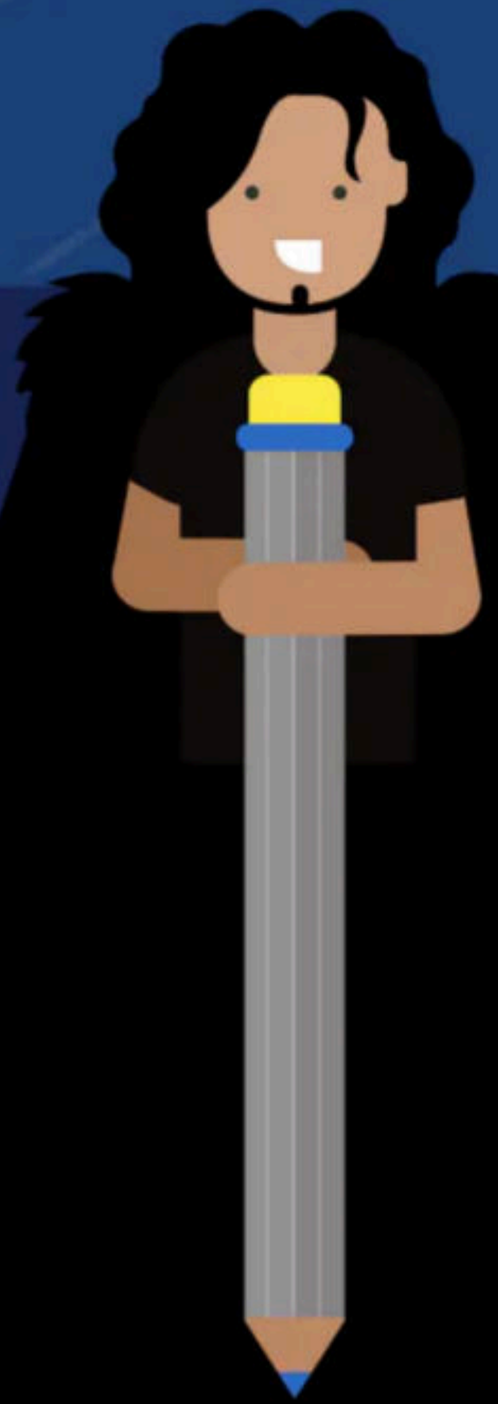
GATE & ESE presents

GATE OF THRONES

Season 2

USE CODE

TO UNLOCK



Get ready for the Final Conquest

65 Questions | 180 Minutes

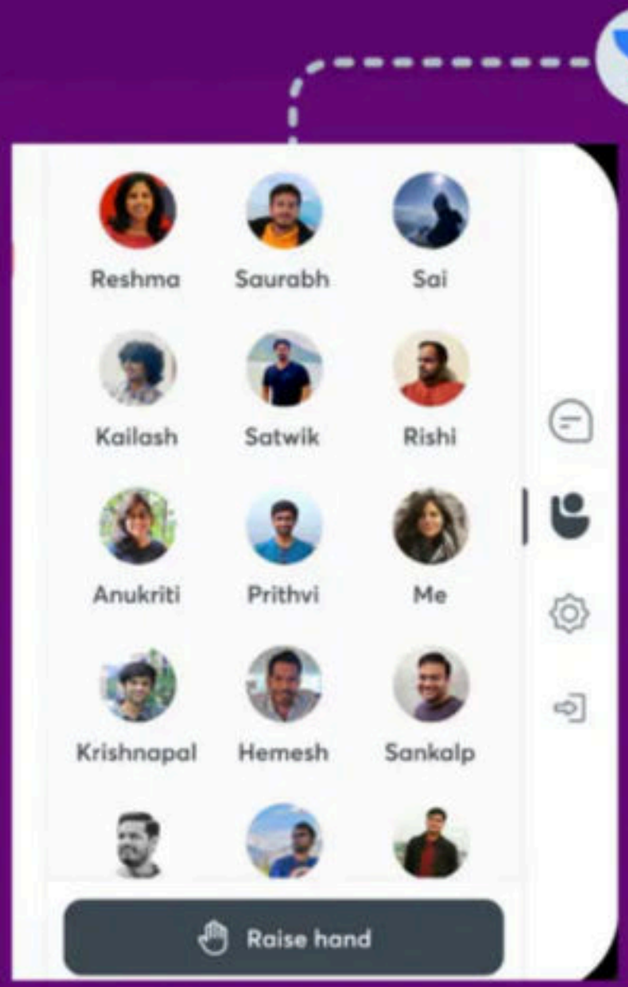
29th October | 6 PM

ME | CE | EE |EC | CS/IT | CH

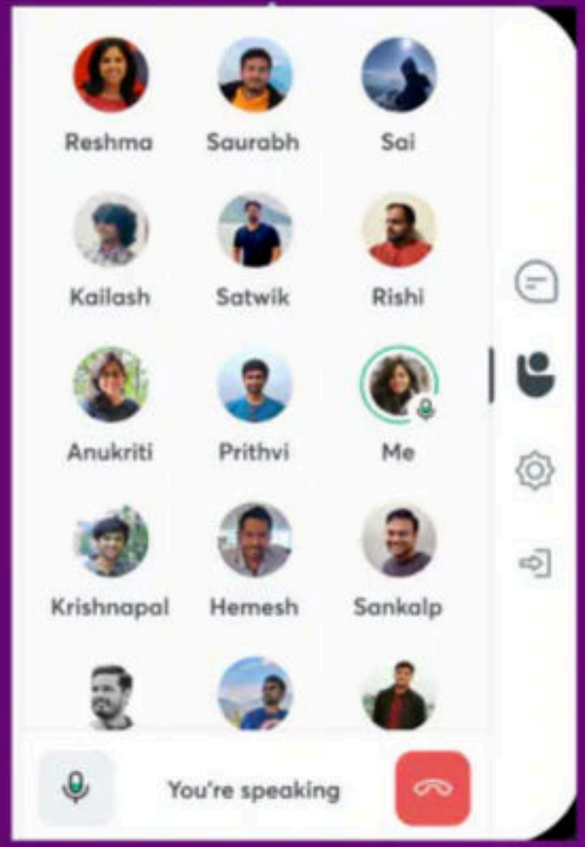
unacademy

RAISE-A-HAND

Talk with your Favourite Educators in live class and clear your doubts!



- Click on the Raise a Hand icon from the vertical tab and click on Raise hand



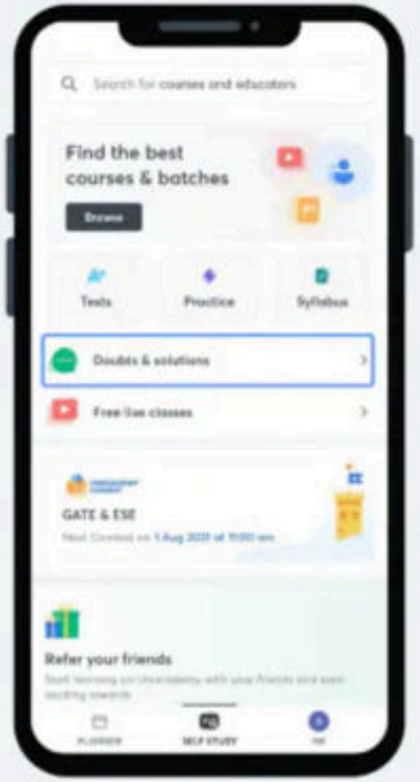
- Talk to the Educator in a Live Class

ASK-A-DOUBT BY UNACADEMY

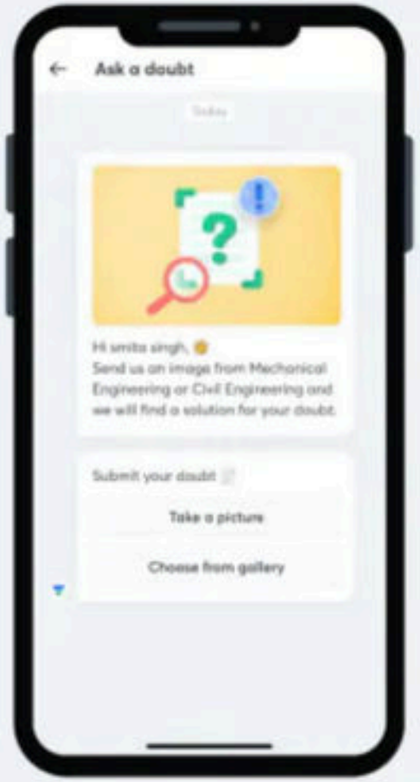
GATE

unacademy

1. Go to the "Doubts & Solutions" feature on your home screen.

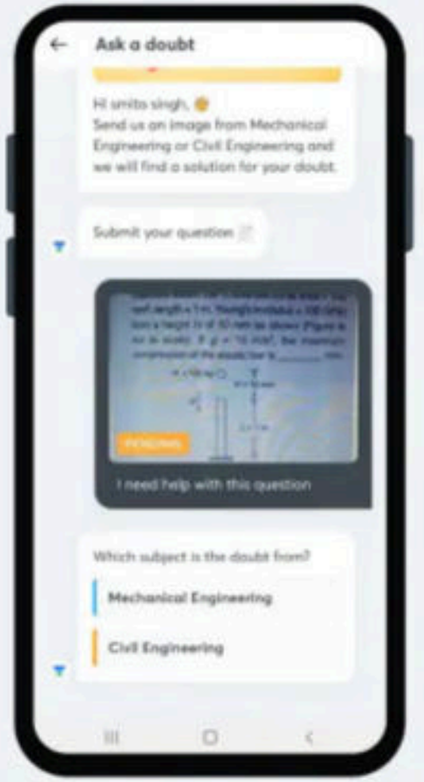


2. Click on "Ask-A-Doubt' & upload your doubt image.



3. Crop the image and highlight the question

4. Choose the subject that the question falls under



unacademy | ICONIC subscription

1:1 Live Mentorship Platform

Let's make your doubt solving easier

Book dedicated 1:1 sessions with your Mentor

- Discuss your Doubts
- Clarify your concepts
- LIVE mentoring sessions for exam strategy

unacademy

CAMPUS PLACEMENT-IT JOBS

BATCHES & TEST SERIES

A to Z: Technical & Aptitude Skills

Weekly Company Specific Mock Tests

Soft Skills & Interview Guidance

***Deloitte Test Series**
Every Saturday (5:00 PM)

HCL Free Test Series
Every Friday (5:00 PM)

Note - *Deloitte Mock Test Series is Available for Plus Subscribers only.
Analysis classes on every Sunday for Deloitte Test Series

New Batches
Every Wednesday

Hall Of Fame Batch
Prepare for Dream Companies such as Deloitte, Samsung, EY etc.

Achievers Batch
Specially curated batches for preparation of TCS, Infosys, Wipro's tests & interviews

Batch Of Titans
Top Educators from Top Categories

Quant & DILR

Ravi Prakash | Udit Saini

Programming

Sanket Singh | Sonia Motwani

DS, CN, OS & DM

Sweta Kumari | Sanchit Jain

Verbal

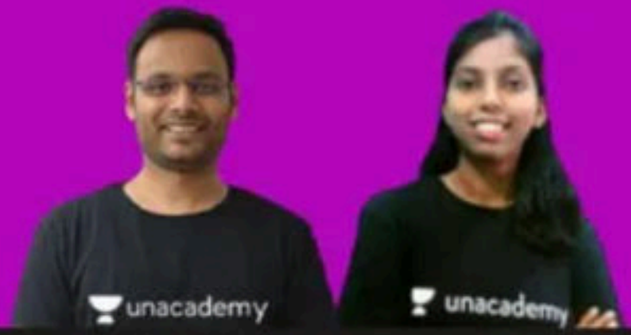
Sameer Ahmad | Shabana

AAROHAN

BATCH FOR GATE 2023

Bilingual | CS & IT | Batch - D

SANCHIT JAIN SUBBA RAO VENKATA RAO M



VISHVADEEP GOTHI SWETA KUMARI

WE START WITH

Digital Electronics

Sanchit Jain

7:00 AM

START DATE

03 NOV 21

GET 10% OFF

USE CODE

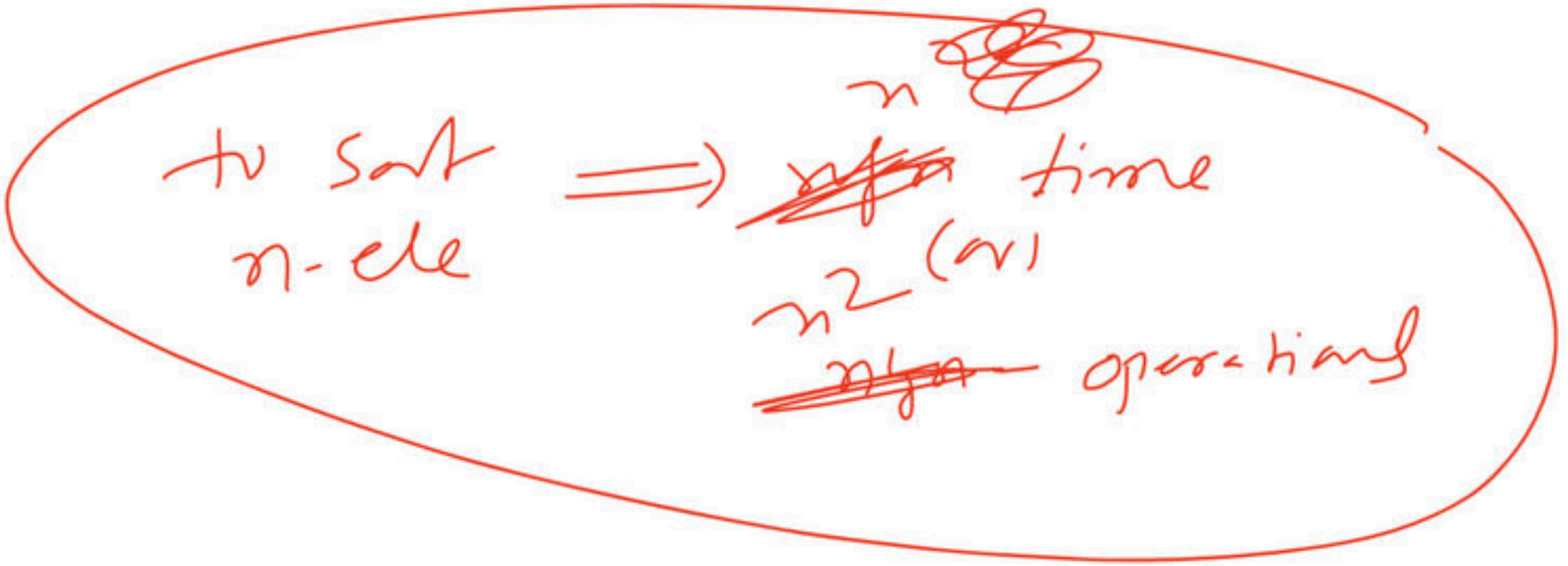
Assume that a merge sort algorithm in the worst case takes 30 seconds for an input of size 64. Which of the following most closely approximates the maximum input size of a problem that can be solved in 6 minutes?

(A) 256

(B) 512

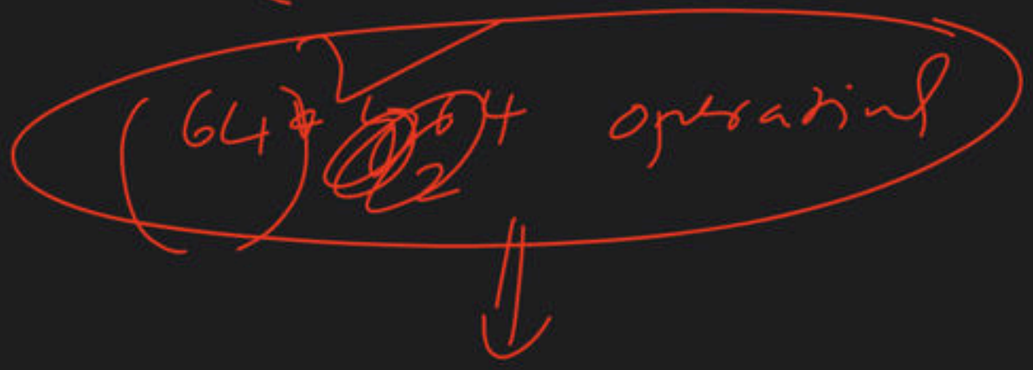
(C) 1024

(D) 2048



64 elements to sort $\Longrightarrow$ 30 sec ✓

(or)



384 operations = 30 sec

1 operation = $\frac{30}{384}$ sec

$\frac{30}{(64)^2}$

Assume x- elements we can sort using merge sort in 6 minutes.

$x \log_2 x$ operations $= 6 * 60$ sec

$x \log_2 x * \frac{6}{384} = 360$ sec

$x \log_2 x = 360 * \frac{384}{30}$ sec

$x = 512$

Consider two strings A = "qpqrr" and B = "pqprqrp". Let x be the length of the longest common subsequence (not necessarily contiguous) between A and B and let y be the number of such longest common subsequences between A and B. Then x + 10y is ---------

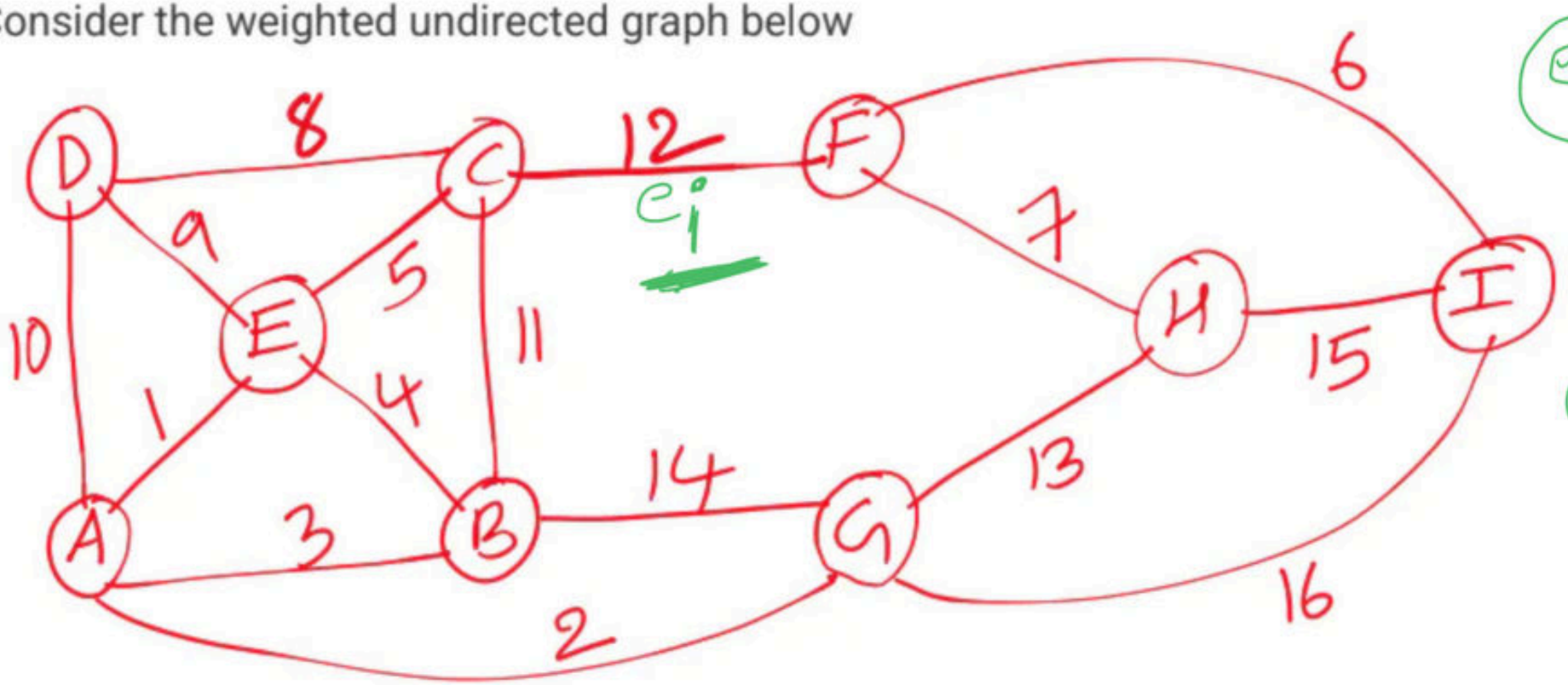
(A) 33

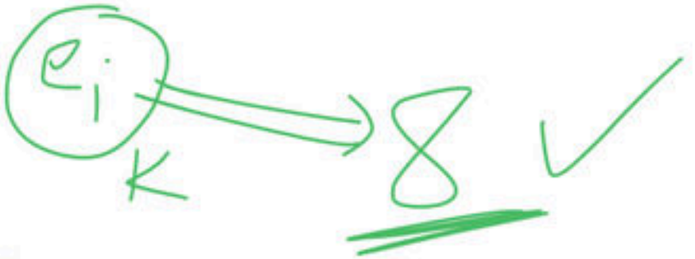
(B) 23

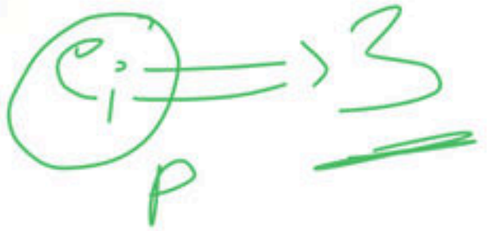
(C) 43

(D) 34

Consider the weighted undirected graph below

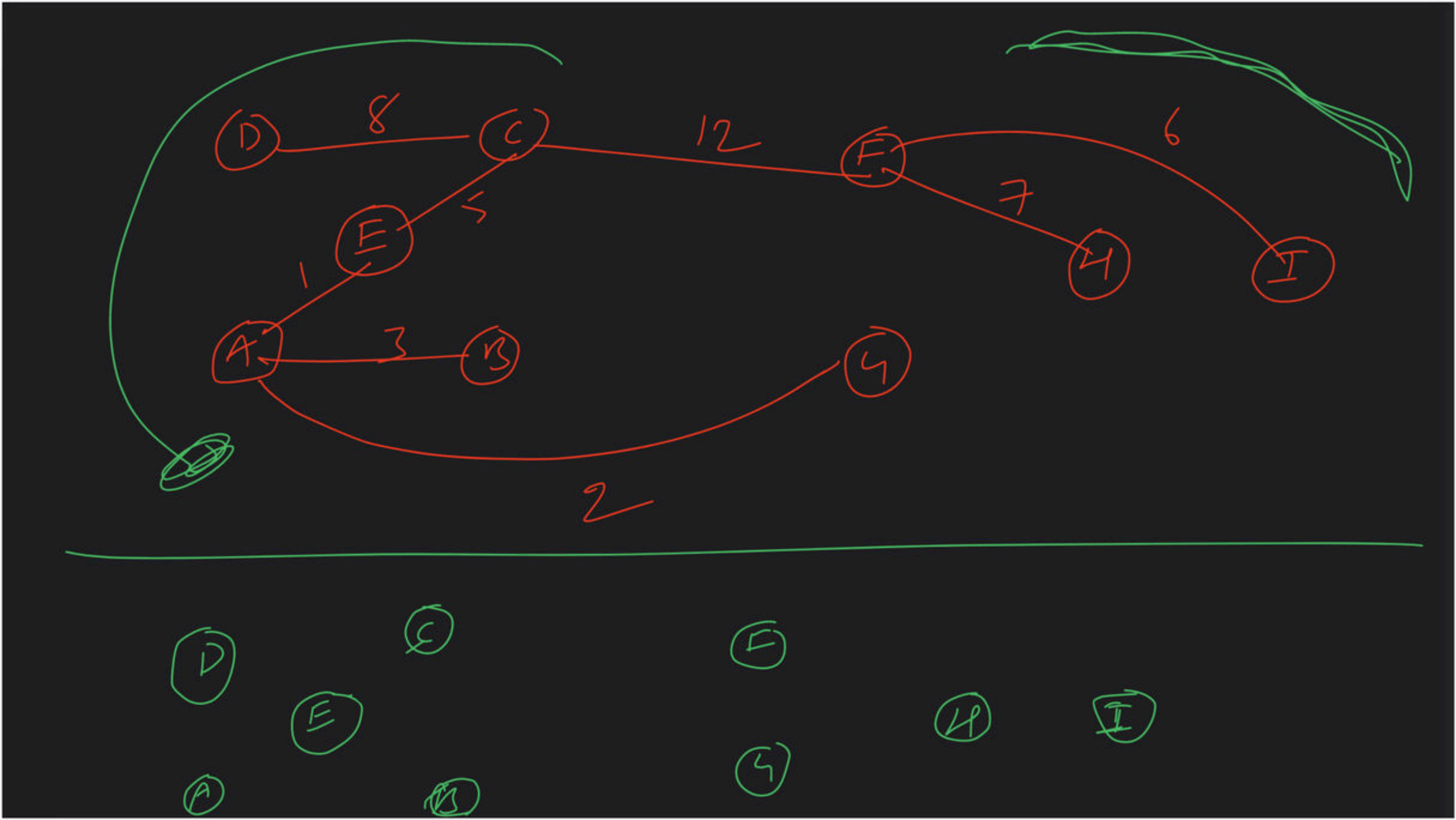


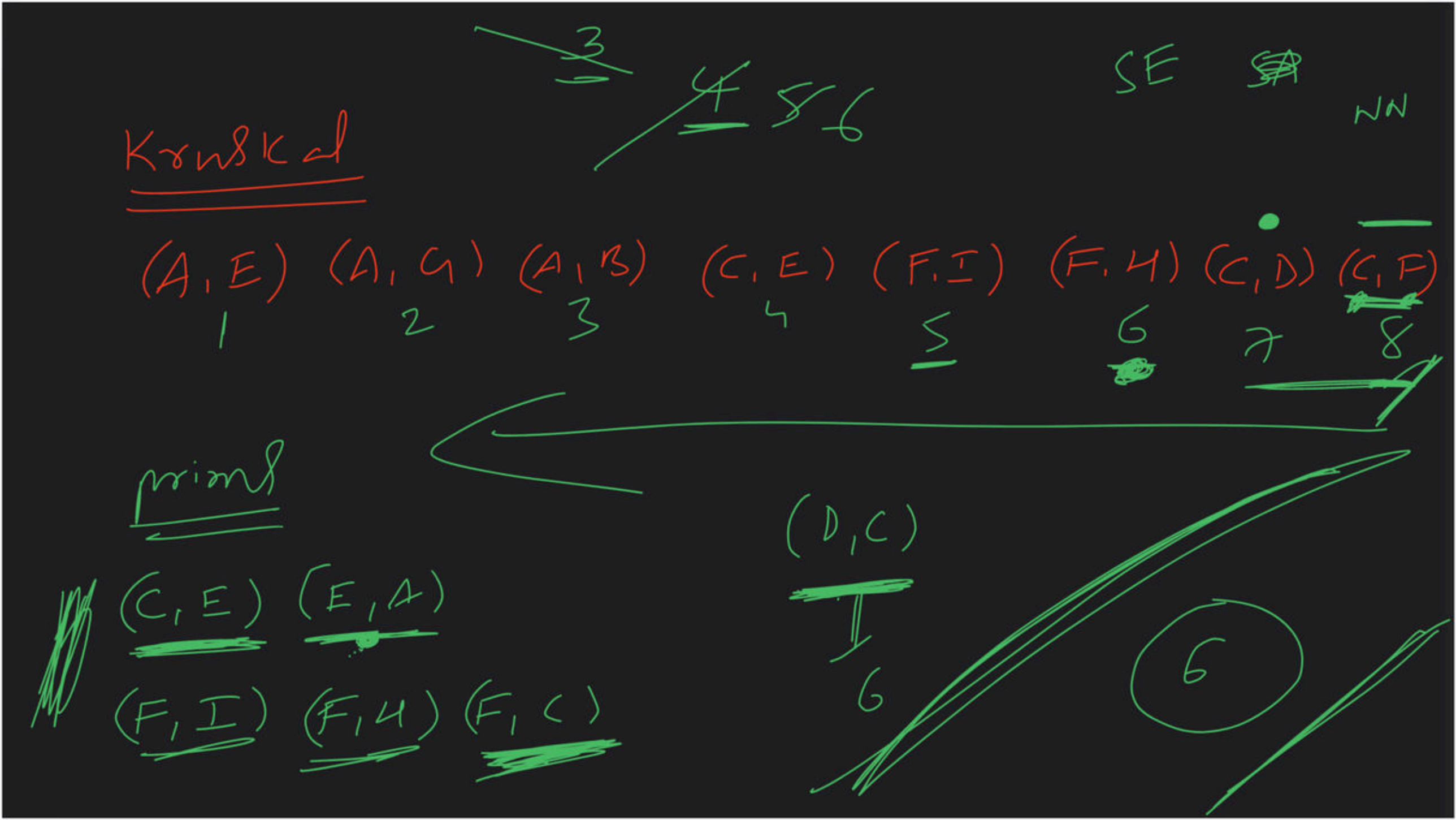




Assume prim's algorithm and kruskal's algorithm are executed on the above graph to find the minimum spanning tree. For a particular edge (e_i) which is included in minimum spanning tree and the position of an edge in minimum spanning tree is denoted by e_{pi} . Where $1 <= e_{pi} <= 8$ (where position defines the order in which edges are included in the MST). Then what is the maximum value of

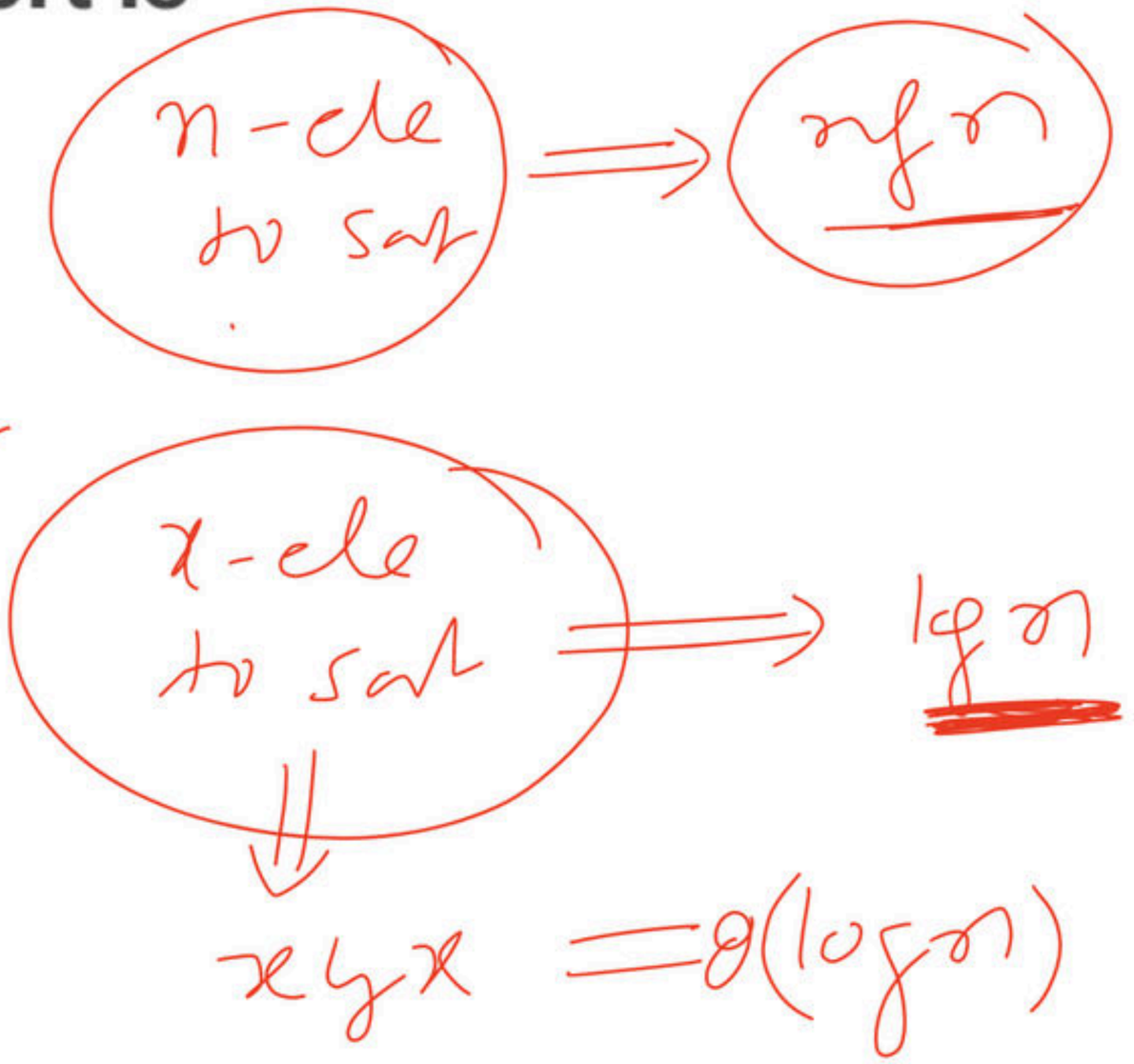
$$|(e_{pi})_{prims} - (e_{pi})_{kruskal}| \; ?$$


D
8
C
12
F
6
E
5
7
H
I
1
A
3
B
G
2
D
C
F
E
H
I
G
A
B


Kruskal
(A, E) (A, G) (A, B) (C, E) (F, I) (F, H) (C, D) (C, F)
1 2 3 4 5 6 7 8
SE
NW
prims
(C, E) (E, A)
(F, I) (F, H) (F, C)
(D, C)
I
6
6

The number of elements that can be sorted in $\Theta(\log n)$ time using heap sort is

1. $\Theta(1)$
2. $\Theta(\sqrt{\log n})$
3. $\Theta(\log n / \log\log n)$
4. $\Theta(\log n)$



$\Rightarrow \Theta(4n)$

$\frac{4n}{44n} \log\left(\frac{4n}{44n}\right)^4$

$\frac{4n}{44n}(44n - 44\log n)$

$4n - \frac{4n * (44n)}{40\log n}$

$\log n$

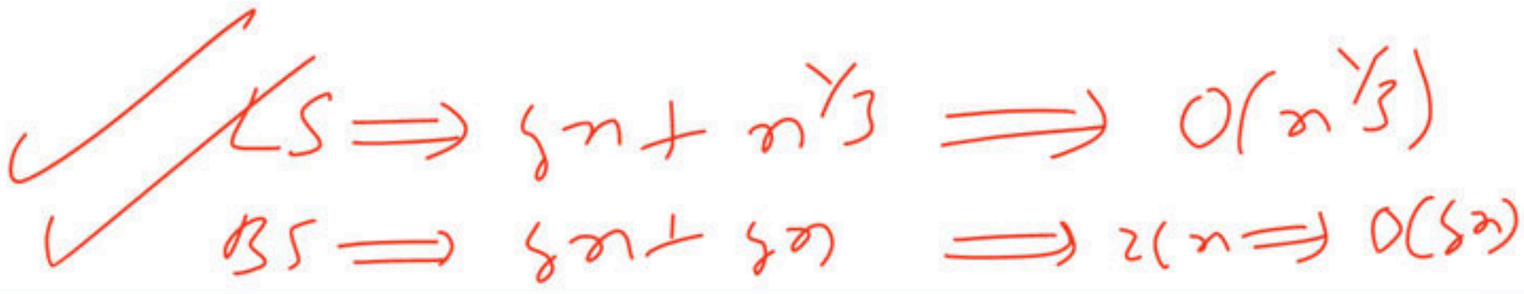
The cube root of a natural number n is defined as the largest natural number m such that $m^3 \leq n$. The complexity of computing the cube root of n (n is represented in binary notation) is:

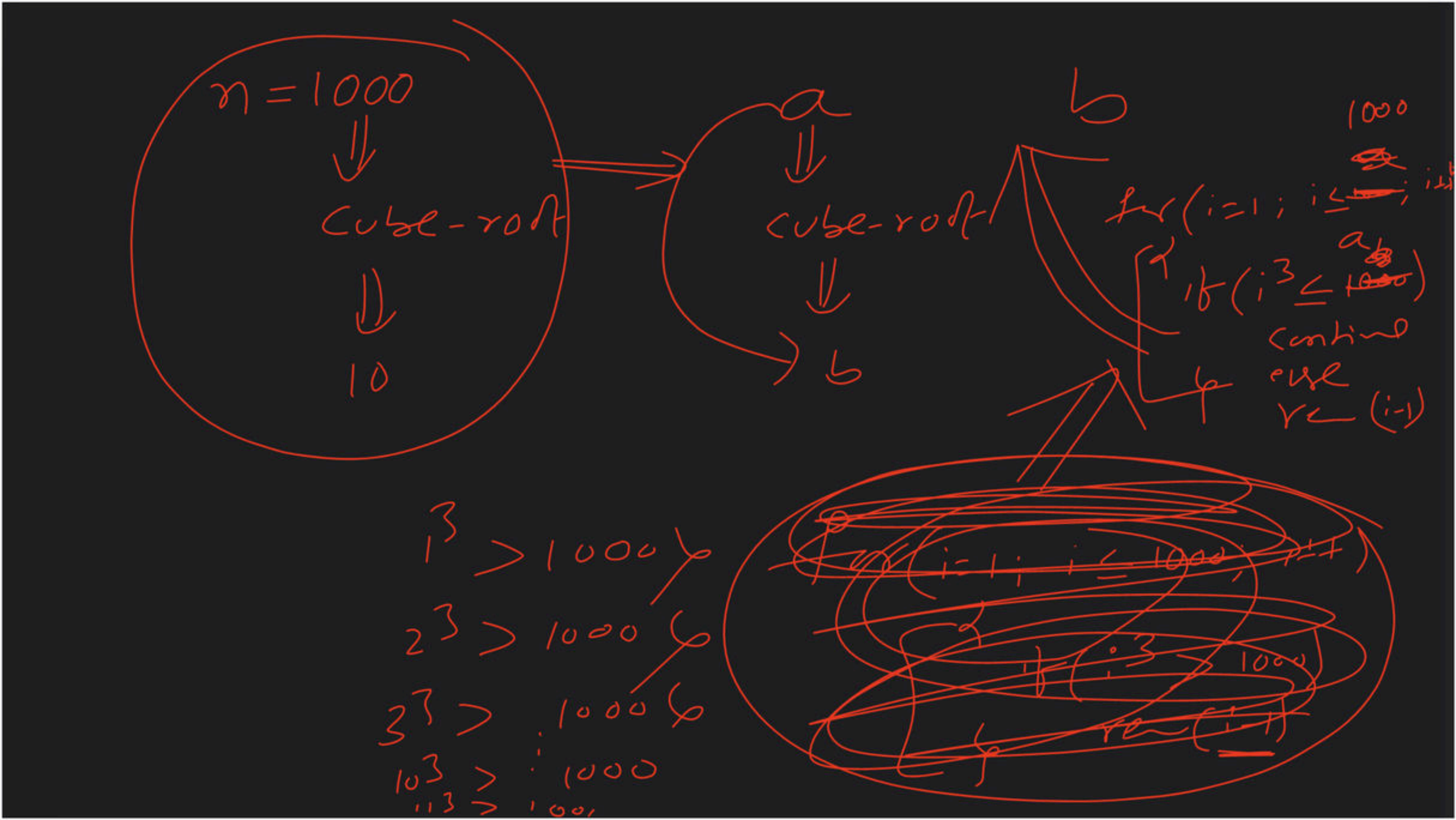
(A) $O(n)$ but not $O(n^{0.5})$

(B) $O(n^{0.5})$ but not $O((\log n)^k)$ for any constant $k > 0$

(C) $O((\log n)^k)$ for some constant $k > 0$, but not $O((\log \log n)^m)$ for any constant $m > 0$

(D) $O((\log \log n)^m)$ for some constant $k > 0.5$, but not $O((\log \log n)^{0.5})$




n = 1000
cube-root
10
a
cube-root
b
b
1000
for(i=1; i≤
a
if(i^3 ≤
continue
else
r ← (i-1)
3
1^3 > 1000 b
2^3 > 1000 b
3^3 > 1000 b
10^3 > 1000
11^3 > 1000
for(i=1; i≤1000; i++)
if(i^3 > 1000)
r ← (i-1)

n

$1^3 \leq n$ ✓

$2^3 \leq n$ ✓

$\vdots$

$(m)^3 \leq n$ ✓

$\vdots$

$O(n^{\frac{1}{3}})$

1000

$(1, 2, 3, \ldots, 499)$ $(500)^3$; $501, 502, \ldots 1000$

$(1\ 2\ 3 -)$ $(250)^3$ $- 500$

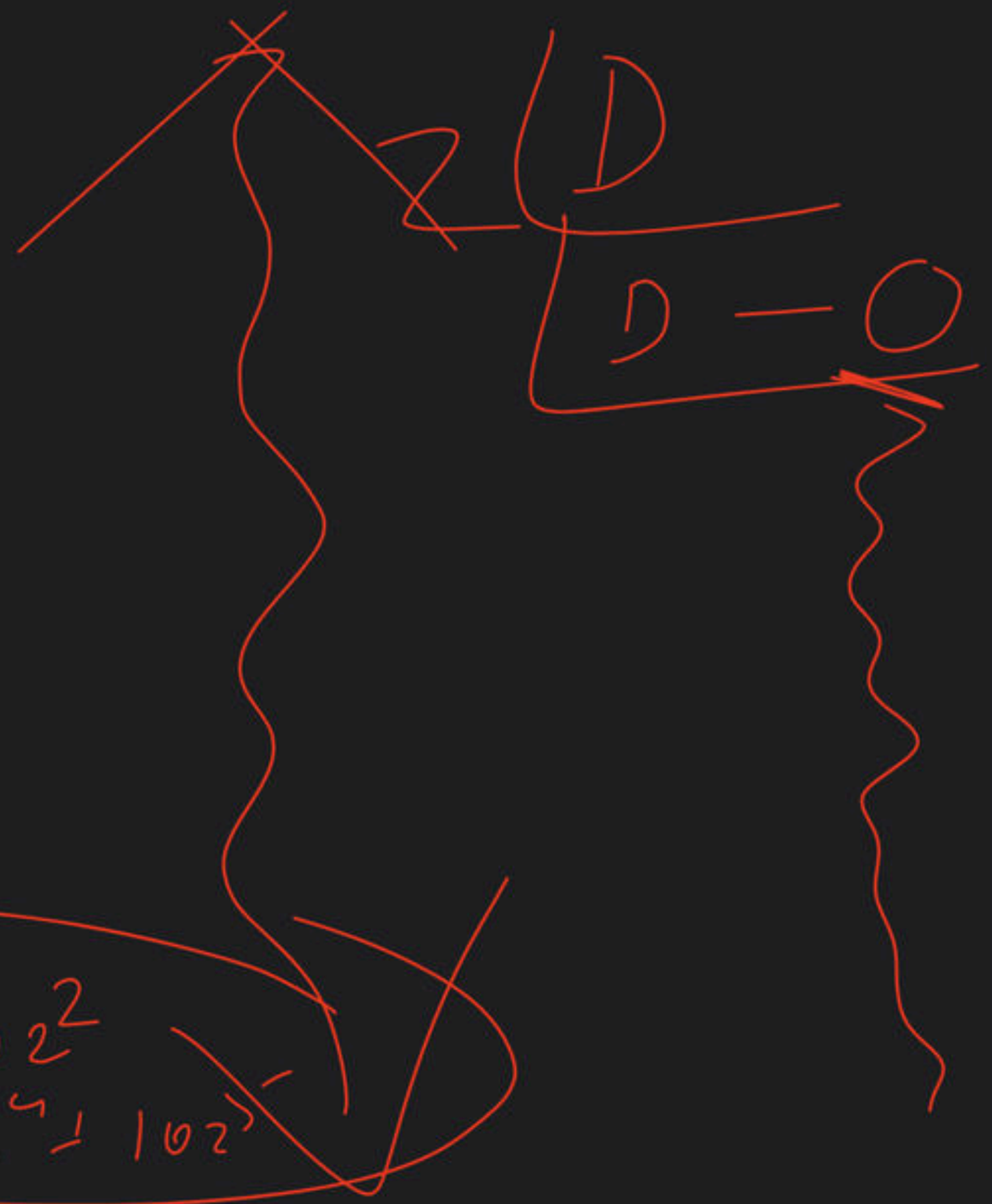
$(1\ 2\ 3 \cdot 2)(-64)^3 \ldots 125$

RS

$4n + O(\log n)$

1010101

$0b2^6 + 1b2^1 + 0b2^2 + 1b2^3 + 0b2^4 + 1b2^5$



900

$(1\ 2\ 3\ -)\ 16^3 \ldots 32$

$1\ 2\ 3\ -\ -\ 8^3 \ldots 16$

$9(-\ -)\ 12^3 \ldots\ldots)$ is

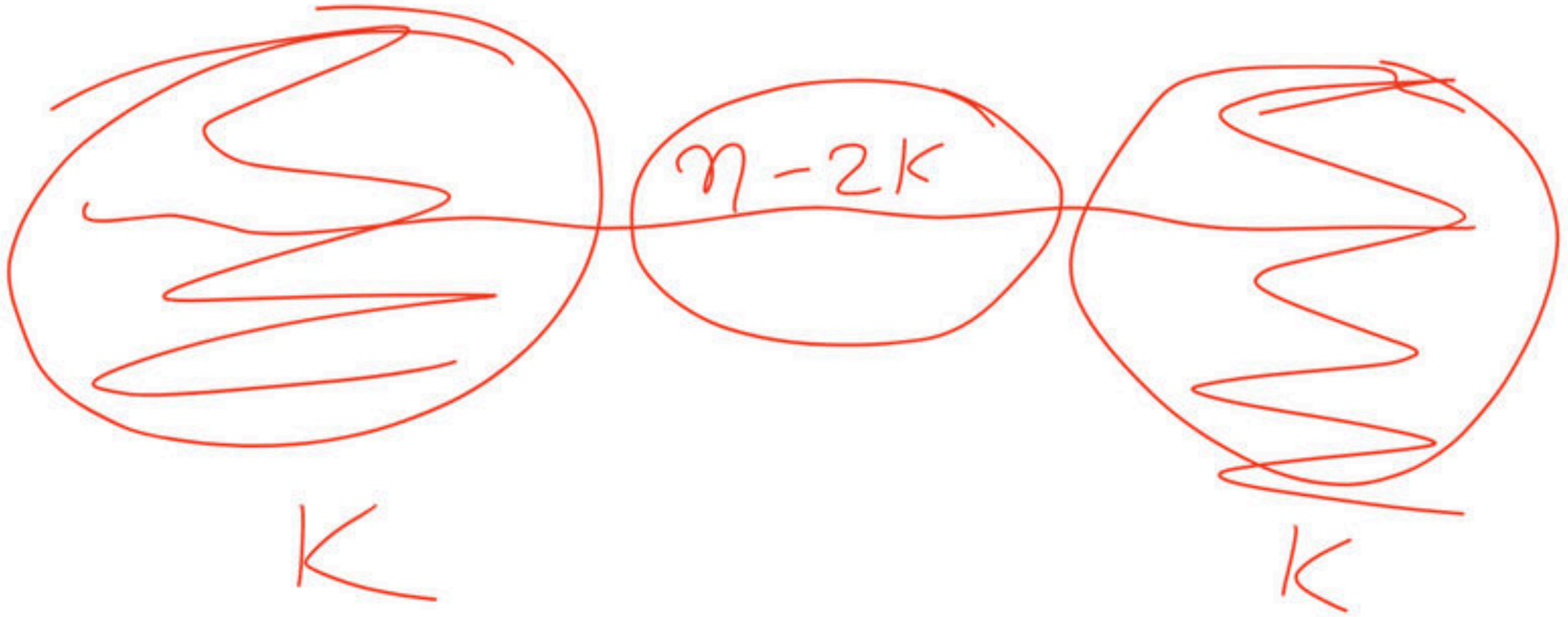
9 10 11

$1^3 \quad 2^3 \quad 3^3 \quad \ldots \quad (n)^3 = n$

```
for (i=1;  i^3 <= n;  i++)
```

An array A of size n is known to be sorted except for the first k elements and the last k elements, where k is a constant. Which of the following algorithms will be the best choice for sorting the array A?

a)Insertion Sort
b)Bubble sort
c)Quicksort
d)Selection sort



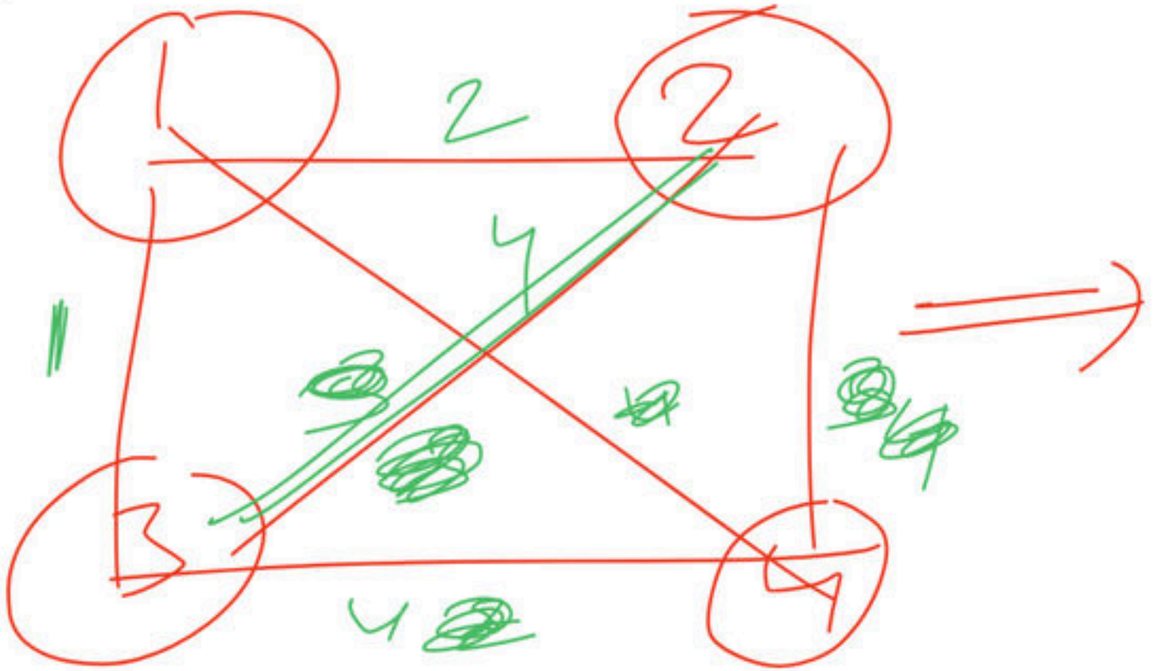
Let G be a complete undirected graph on 4 vertices, having 6 edges with weights being 1, 2, 3, 4, 5, and 6. The maximum possible weight that a minimum weight spanning tree of G can have is.

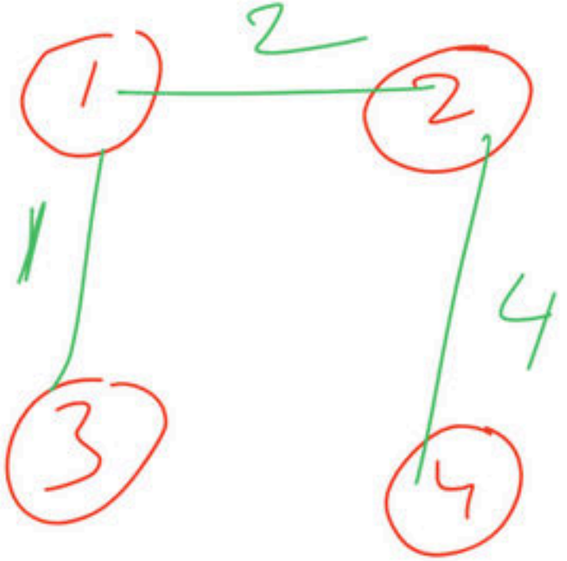
(A) 6

(B) 7

(C) 8

(D) 9





An undirected graph G (V, E) contains n (n > 2) nodes named v_1 , v_2 ,...,v_n. Two nodes v_i and v_j are connected if and only if $0 < |i-j| \leq 2$. Each edge (v_i , v_j) is assigned a weight i+j. The cost of the minimum spanning tree of such a graph with 7 nodes is -------

For parameters a and b, both of which are $\omega(1)$, $T(n)=T(n^{1/a})+1$, and $T(b)=1$. Then T(n) is-----

A. $\Theta(\log_a \log_b n)$
B. $\Theta(\log_{ab} n)$
C. $\Theta(\log_b \log_a n)$
D. $\Theta(\log_2 \log_2 n)$

There are n unsorted arrays: $A_1, A_2, \ldots, A_n$. Assume that n is odd. Each of $A_1, A_2, \ldots, A_n$ contains n distinct elements. There are no common elements between any two arrays. The worst-case time complexity of computing the median of the medians of $A_1, A_2, \ldots, A_n$ is----

A. $O(n)$
B. $O(n\log n)$
C. $O(n^2)$
D. $\Omega(n^2\log n)$

There are 2 sorted arrays A and B of size n each, to find the median of the array obtained after merging the above 2 arrays(i.e. array of length 2n). Then what should be the complexity?

Ex:
ar1[] = {1, 12, 15, 26, 38}
ar2[] = {2, 13, 17, 30, 45}

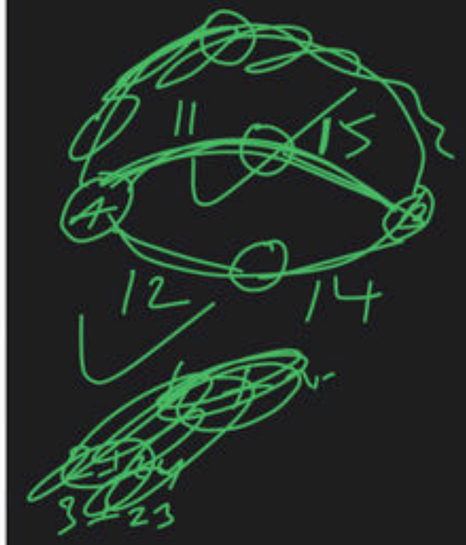
Thank you

Hallo

unacademy

▲ 1 • Asked by Nayan

Sir plz discuss..



If we use Radix Sort to sort n integers in the range $(n^{k/2}, n^k]$, for some k>0 which is independent of n, the time taken would be?

(A) $\Theta(n)$

(B) $\Theta(kn)$

(C) $\Theta(n\log n)$

(D) $\Theta(n^2)$

Following algorithm(s) can be used to sort n in the range $[1 \ldots n^3]$ in $O(n)$ time

1. Heap sort
2. Quick sort
3. Merge sort
4. Radix sort
5. Counting sort
6. Selection sort

An articulation point in a connected graph is a vertex such that removing the vertex and its incident edges disconnects the graph into two or more connected components. Let T be a DFS tree obtained by doing DFS in a connected undirected graph G. Which of the following options is/are correct?

A. If u is an articulation point in G then for every pair (x,y) where x is an ancestor of u in T and y is a descendent of u in T all paths from x to y in G must pass through u.

B. If u is an articulation point in G then there exist at least one pair (x,y) where x is an ancestor of u in T and y is a descendent of u in T, then all paths from x to y in G must pass through u.

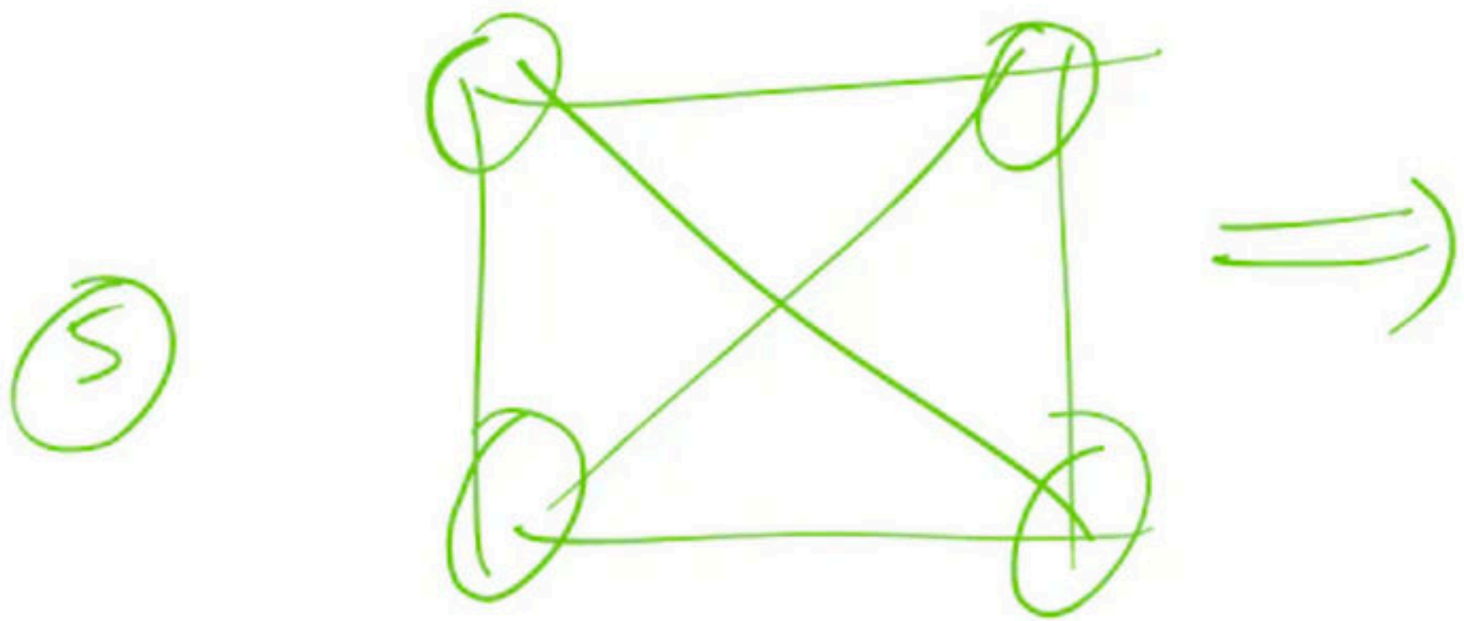
C. If u is an articulation point in G then there exist at most one pair (x,y) where x is an ancestor of u in T and y is a descendent of u in T, then all paths from x to y in G must pass through u

D. If u is an articulation point in G then there exist exactly one pair (x,y) where x is an ancestor of u in T and y is a descendent of u in T, then all paths from x to y in G must pass through u

0101101111001010 = 10100101
01100111
1111001110 1111001111

ms = 0 3 8

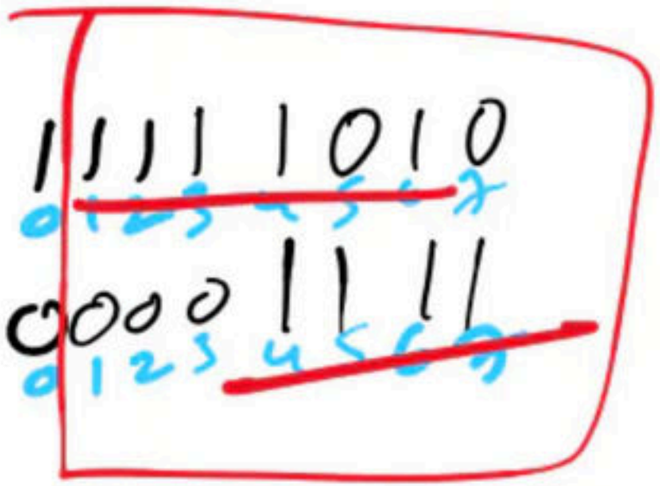
s = 0 1 2 3 0 1 2 8 0


S

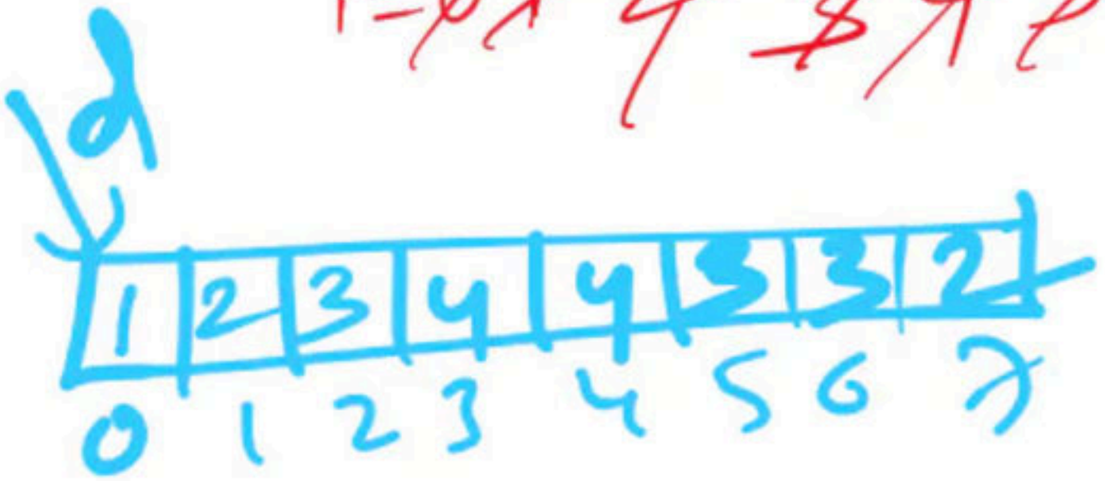
Given two arrays of numbers a1, a2, a3,...an and b1, b2, .. bn where each number is 0 or 1, the fastest algorithm to find the largest span(i, j) such that ai + ai+1 + ai+2.....aj = bi +, bi+1, bi+3... bj . or report that there is not such span.

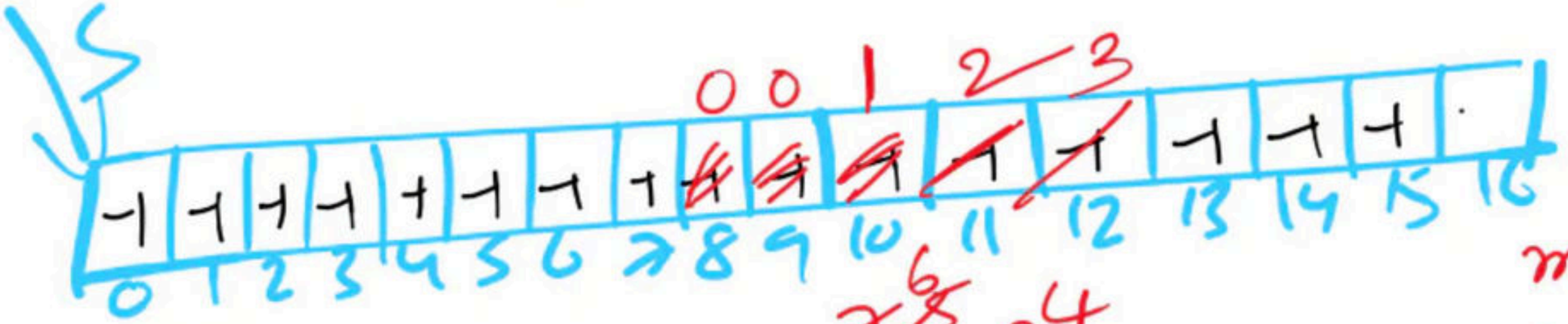
(a) Takes O (n^3) and Ω (2^n) time if hashing is permitted
(b) Takes O (n^3) and Ω (n^2.5) time in the key comparison model
(c) Takes θ (n) time and space
(d) Takes O (√n) time only if the sum of the 2n elements is an even number

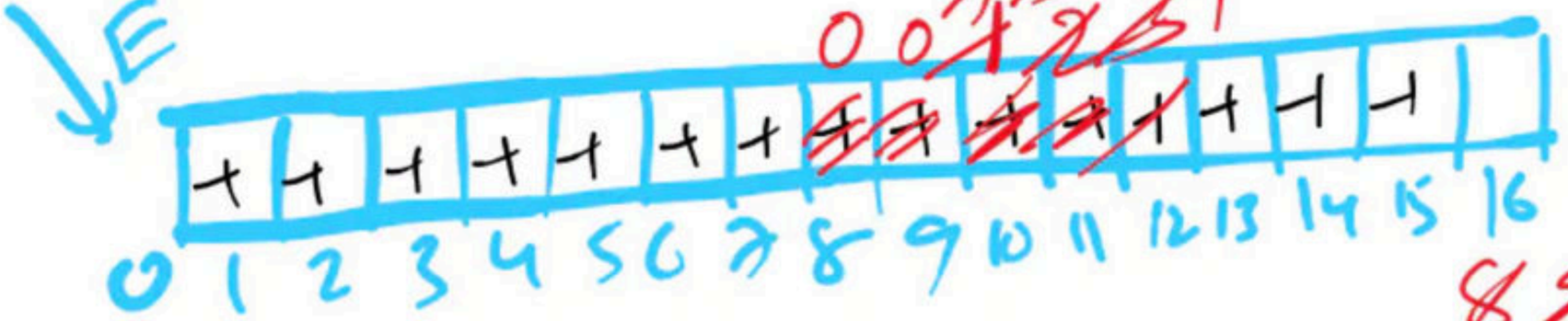
n=8



i=0 1 2 3 4 5 6 7

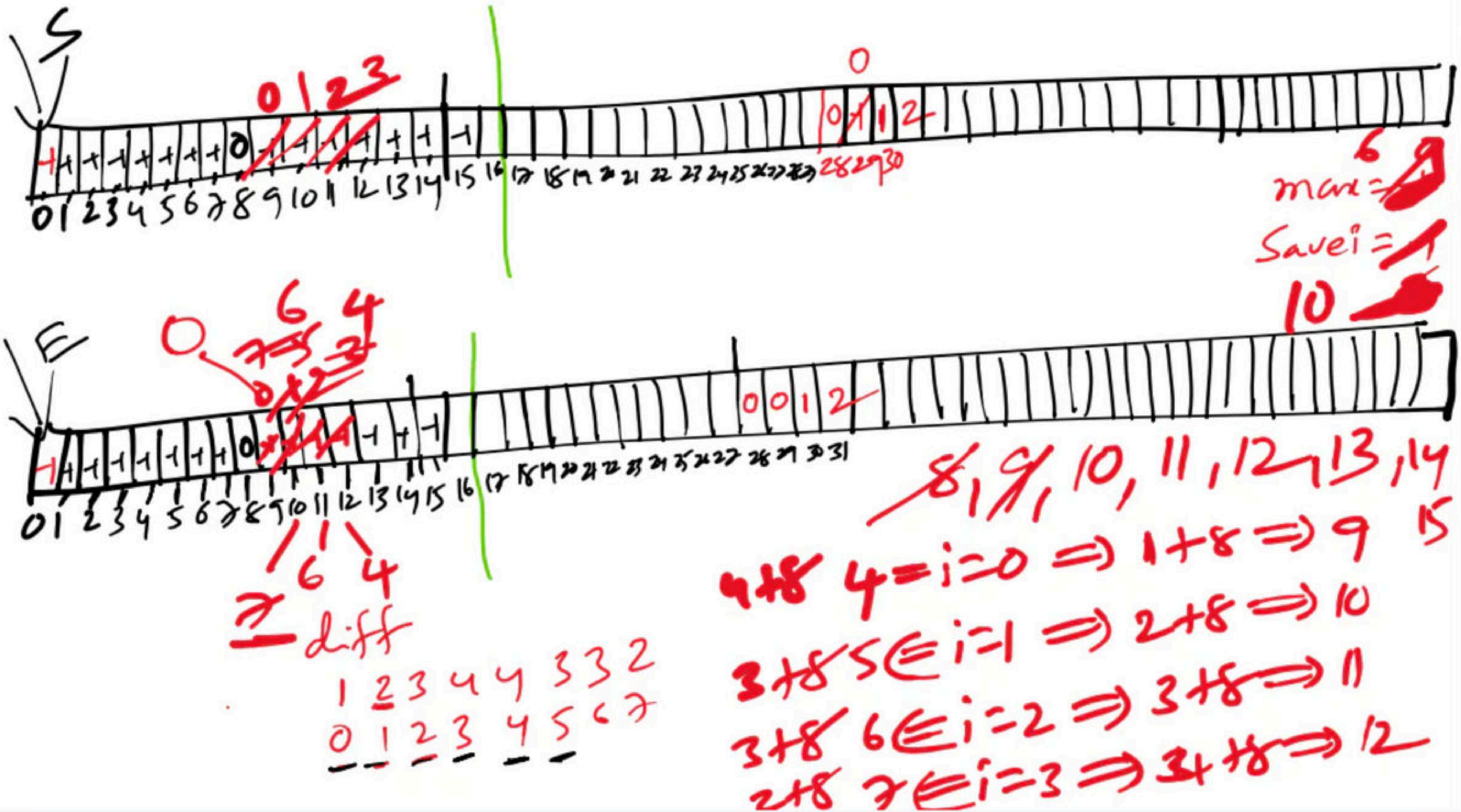


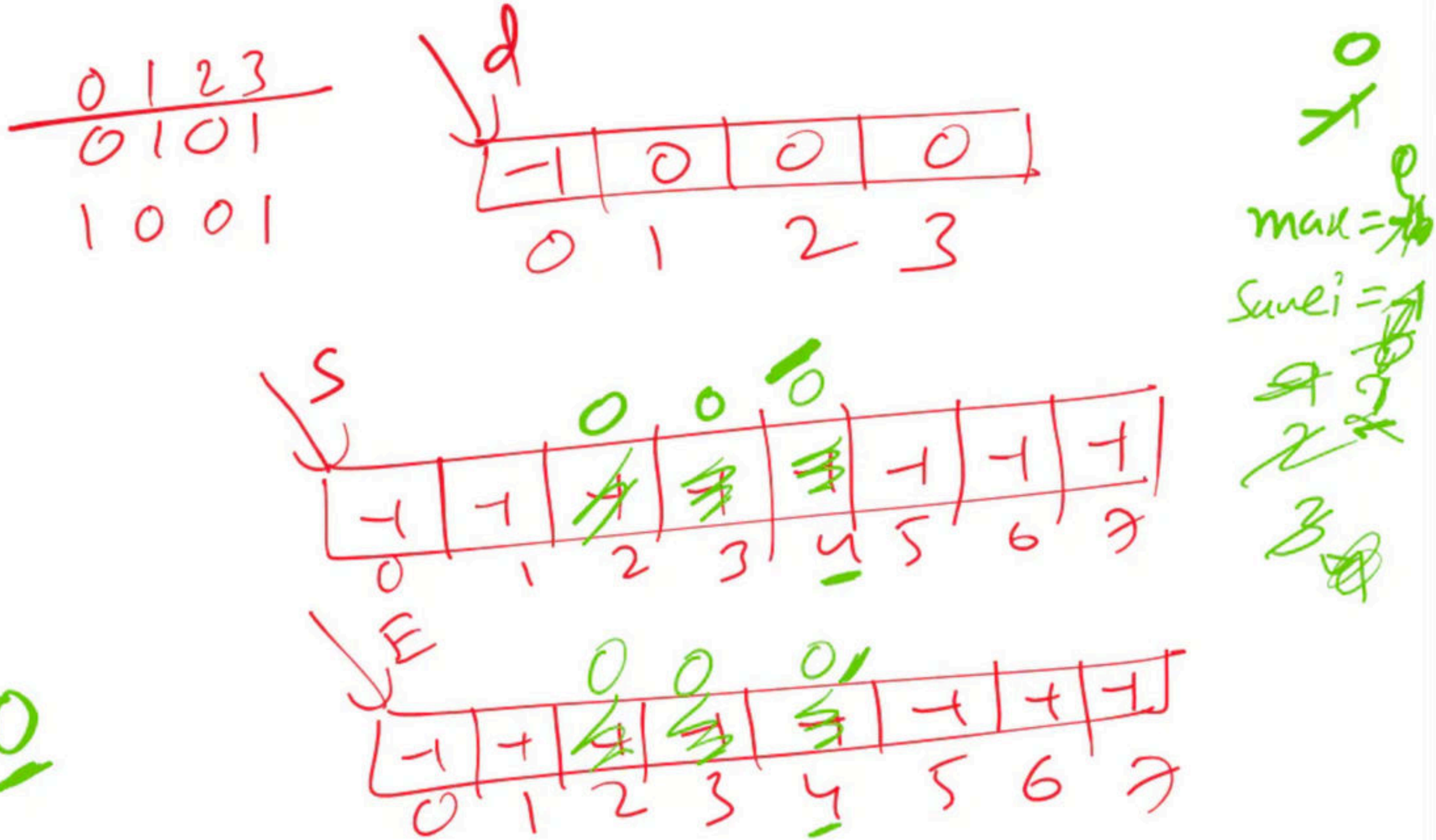




6
0
max=-1
Savei=1
6
10

8 9 10 11 12


S
E
max =
Savei =
diff
4 = i = 0 ⇒ 1+8 ⇒ 9
5 ∈ i = 1 ⇒ 2+8 ⇒ 10
6 ∈ i = 2 ⇒ 3+8 ⇒ 11
7 ∈ i = 3 ⇒ 4+8 ⇒ 12


0 1 2 3
0 1 0 1
1 0 0 1
d
-1 0 0 0
0 1 2 3
max =
sum =
S
-1 -1 -1 -1 -1 -1 -1 -1
0 1 2 3 4 5 6 7
E
-1 -1 -1 -1 -1 -1 -1 -1
0 1 2 3 4 5 6 7