上 海 交 通 大 学 试 卷

(2023~ 2024~1 Academic Year/Fall Semester)

Class No.______ECE2810J______Name in English or Pinyin: ____________________

Student ID No._______________ Name in Hanzi (if applicable): ____________________

ECE2810J Data Structures and Algorithms

Final Exam

12/14/2023 10:00-11:40

The exam paper has 18 pages in total.

You are to abide by the University of Michigan-Shanghai Jiao Tong University Joint Institute (UM-SJTU JI) honor code. Please sign below to signify that you have kept the honor code pledge.

THE UM-SJTU JI HONOR CODE

I accept the letter and spirit of the honor code:

I have neither given nor received unauthorized aid on this examination, nor have I concealed any violations of the Honor Code by myself or others.

Signature:____________

Please enter grades here:

Exercises No. 题号	**Points** 得分	**Grader's Signature** 流水批阅人签名
1		
2.1		
2.2		
2.3		
2.4		
..		
..		
..		
..		
..		
Total 总分		

1 Multiple Choice Questions (40 Points)

ANSWER SHEET FOR MULTIPLE CHOICE QUESTIONS:
Please fill the correct answers in the **answer sheet**. Answers outside the answer sheet will **NOT** be graded.

1	2	3	4	5	6
B	B	C	A	A	D
7	8	9	10	11	12
A	B	CE	AC	CD	ABD

Question 1-8 have exactly ONE correct answer. Each question is worth 3 points.

1. Jack is working on a project where he implemented using a $\Theta(n^2)$ algorithm. When he ran his own test file on the program, however, he noticed that the program ran in $\Theta(n)$ time. Which of the following is **NOT** a possible reason?

   A) Jack incorrectly analyzed the time complexity of the algorithm.
   B) Jack's test input exposed worst-case behavior.
   C) Jack's test case input size was too small.
   D) None of the above.

2. Consider an unsorted array of size n containing unique numbers from 0 to n. One number is missing from the array. What is the worst-case time and memory complexity of finding the missing number using the most efficient algorithm?

   A) Time: $\Theta(\log n)$, Memory: $\Theta(1)$
   B) Time: $\Theta(n)$, Memory: $\Theta(1)$
   C) Time: $\Theta(n)$, Memory: $\Theta(n)$
   D) Time: $\Theta(n^2)$, Memory: $\Theta(1)$

3. Given a hash table with size $M = 7$ and hash function $H(n) = (2n + 1) \bmod M$. Collisions are resolved using **linear** probing. What would the table look like (with an X denoting an empty location after inserting the following elements in the order given?

   3, 26, 8, 14, 19, 98

   A) 3, 14, X, 8, 26, 98, 19
   B) X, 3, 14, 98, 8, 19, 26
   C) 3, 14, 98, 8, 26, 19, X
   D) X, 3, 98, 14, 8, 26, 19

4. Which of the following represents a valid binary max-heap? The "top" of the heap is the element at the first index of the array.

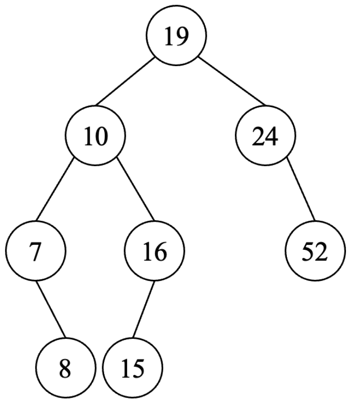
   A) 24, 20, 8, 15, 8, 7, 3

B) 33, 17, 19, 18, 4, 17, 2

C) 17, 6, 17, 5, 3, 18, 4

D) 4, 7, 9, 22, 13, 19, 26

5. Given the following AVL tree:



Perform the following operations:

1. Delete 19 and replace with its in-order successor.
2. Insert 12.

What are the left and right children of "24" after performing these two operations?

Hint: *if the in-order traversal of a binary tree is ABC, then C is the in-order successor of B.*

A) 15, 52

B) 12, 52

C) 16, 52

D) `nullptr`, 52

6. Which of the following algorithms cannot be optimized using a heap?

A) Prim's algorithm

B) Dijkstra's algorithm

C) Selection sort

D) Topological sort

7. Suppose we want to search for a point at $(2, 1)$ in a 2D grid from the origin $(0, 0)$ using BFS. For each point searched, we can search for all the four adjacent points: $(x+1, y)$, $(x-1, y)$, $(x, y+1)$, $(x, y-1)$, in arbitrary order. What **CANNOT** be the possible number of points visited when we find the target point? Both the origin and the target point are included.

A) 13

B) 17

C) 21

D) 25

8. Suppose we want to build a Trie that contains the words in the set {apple, app, ape, bat, batch, batman, car, cat, cart, card}, then how many edges are there in the Trie?

   Hint: *use "$" to indicate the end when a string in the set is exactly a prefix of another string.*

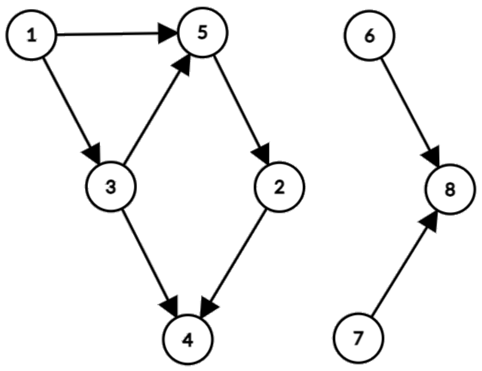
   A) 18
   B) 19
   C) 20
   D) 21

Questions 9-12 have MULTIPLE correct answers. Each question is worth 4 points. You will get 2 points if your answer contains part of the correct options. No points will be given if you leave it empty or your answer contains at least one WRONG option.
Example: if the correct answer is ABC, then answers like "AB" and "C" will be given 2 points, but no points will be given to answers like "AD" and "ABCD".

9. A student is given an array of n integer elements that needs to be sorted. He wants the execution time of the sorting algorithm to be no more than $O(n \log n)$, **regardless of the initial order** of the array. Which of the following algorithms can meet this requirement?

   A) Insertion sort
   B) Selection sort
   C) Merge sort
   D) Quick sort
   E) Heap sort

10. Given a unweighted directed graph $\mathcal{G}$:



   Which of the following sequences are valid topological order of $\mathcal{G}$?

   A) 1, 3, 7, 5, 6, 2, 8, 4
   B) 6, 7, 1, 8, 3, 4, 5, 2
   C) 1, 3, 5, 2, 4, 6, 7, 8

D) 1, 3, 6, 7, 2, 4, 5, 8

11. Which of the following statements are **correct** about topological sort?

    A) There exists a DAG that has no topological order.

    B) There are at most $\frac{n(n+1)}{2}$ topological orders for any DAG with n vertices

    C) The total runtime to get a topological order for a dense DAG is $O(|V|^2)$.

    D) The total runtime to get a topological order for a sparse DAG is $O(|V|)$.

12. In the following statements about k-d tree operations, which of the statements are **NOT** correct?

    A) k-d tree is a balanced binary tree, which guarantees the time complexity of operations limited to $O(\log n)$.

    B) When deleting a non-leaf node, we should always select the node with the minimum value from the right subtree to replace the deleted node.

    C) The worst-case time complexity for performing a range search in a k-d tree is $O(n)$.

    D) The average time complexity for inserting a new node in a k-d tree is $O(k \log n)$, where k is the number of dimensions.

2 Writing Questions (60 Points)

2.1 K-D Tree & Hash Table (14 Points)

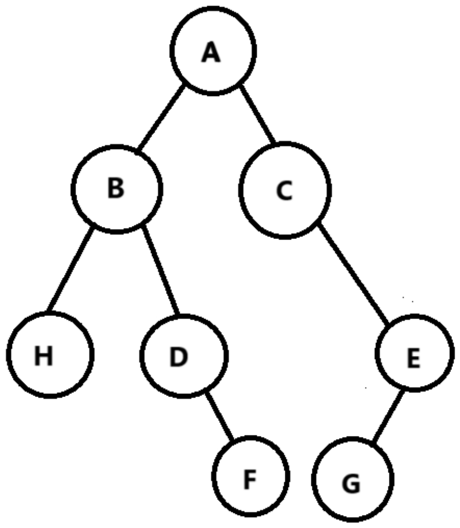
1) (5 points) Suppose that you have received a report for a English proficiency test containing the final scores of 8 students (A to H), and you want to use a **K-D tree** to store these data. The following is the test report:

#	Total	Listening	Reading	Speaking	Writing
A	92	26	24	19	23
B	102	25	23	28	26
C	90	29	21	21	19
D	94	21	27	20	26
E	116	30	29	28	29
F	90	15	25	23	27
G	91	27	24	18	22
H	85	20	18	22	25

Please insert those data in the given order (from A to H) and draw the final K-D tree, following the method discussed in the lecture. The dimension form should be **(Listening, Reading, Speaking, Writing)**.

You are highly recommended to simply ***use A-H*** *to represent each node in the K-D tree.* ***Directly showing the final result*** *would be enough.*

Solution:



2) (9 points) You are given a hash table with the following properties:

- The table size is $M = 7$ and the hash function is $H(n) = n \bmod 7$.
- Collisions are resolved using **quadratic** probing.
- The maximum permissible load factor is $L_{max} = 0.7$.

Now you are required to insert 6 elements into this hash table in order:

13, 15, 24, 6, 23, 40.

i) What would the table look like right after inserting the first 5 elements? (3 points)

ii) After inserting 5 elements, the load factor has exceeded L_{max}, which means a **rehashing operation** is required. Suppose that the new table has size $M' = 17$ with the new hash function $H'(n) = n \bmod 17$. What would the table look like after inserting all those 6 elements? (3 points)

iii) The single operation of rehashing is time-consuming. However, the actual time complexity of insertion for the hash table involving rehashing is not so bad. Can you briefly explain why? (No calculation is needed in this question.) (3 points)

Solution:

i) Write your answer in the following table:

0	1	2	3	4	5	6
6	15	23	24			13

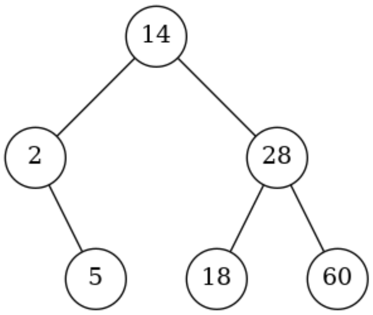
ii) Write your answer in the following table:

0	1	2	3	4	5	6	7	8
						6	23	24
9	10	11	12	13	14	15	16	/
	40			13		15		/

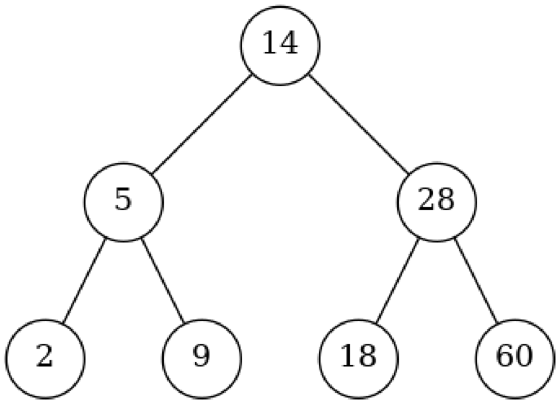
iii) The rehashing operation does not happen frequently. Applying the amortized analysis, the average time complexity of insertion is still $O(1)$.

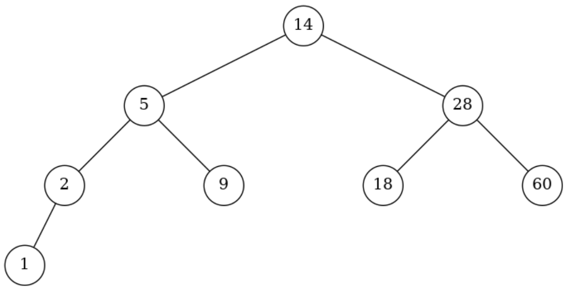
2.2 AVL Tree (14 points)

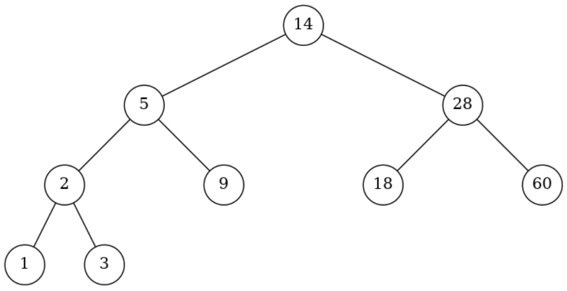
1. (4 points) Insert $9, 1, 3, 4$ one by one into the AVL tree shown below. Draw the AVL trees after each insertion. (4 AVL trees in total)

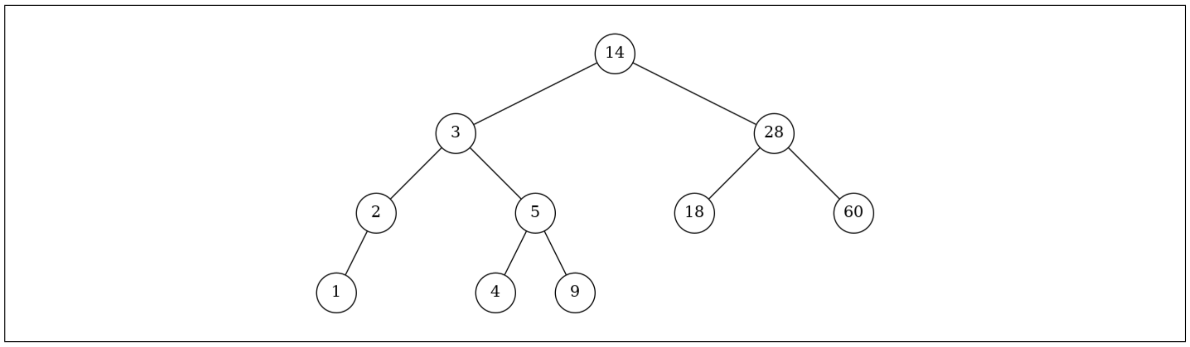


Solution:









2. (2 points) For a search operation, compare the worst-case time complexity of an AVL Tree of n nodes with the worst-case time complexity of a BST (Binary Search Tree) of n nodes.

Solution:
AVL:$O(\log(n))$ < BST:$O(n)$ (2')

3. (2 points) What's the time complexity of traversing an AVL tree of n nodes? Give the answer and explanation.

Solution:

Traversing through n nodes(1') requires $O(n)$(1') time complexity

4. (3 points) What's the minimum number of nodes in an AVL tree of height 3? What's the answer for an AVL tree of height H?
Hint: Fibonacci number $f_n = \frac{1}{\sqrt{5}}\left[\left(\frac{1+\sqrt{5}}{2}\right)^n - \left(\frac{1-\sqrt{5}}{2}\right)^n\right]$

Solution:
The minimum number of nodes in an AVL tree of height 3 is 7.(1')
Suppose a_n represents the minimum number of nodes in an AVL tree of height n. Then,

$$a_n = a_{n-1} + a_{n-2} + 1$$
$$a_n + 1 = (a_{n-1} + 1) + (a_{n-2} + 1)$$

Let $b_n = a_n + 1$, hence

$$b_0 = 2 = f_3$$
$$b_1 = 3 = f_4$$
$$b_n = b_{n-1} + b_{n-2}(1')$$

Using the property of Fibonacci number:

$$b_H = f_{H+3} = \frac{1}{\sqrt{5}}[(\frac{1+\sqrt{5}}{2})^{H+3} - (\frac{1-\sqrt{5}}{2})^{H+3}]$$
$$a_H = f_{H+3} - 1 = \frac{1}{\sqrt{5}}[(\frac{1+\sqrt{5}}{2})^{H+3} - (\frac{1-\sqrt{5}}{2})^{H+3}] - 1(1')$$

5. (3 points) Based on the results obtained in (4), prove the worst-case time complexity of a search operation on an AVL tree of n nodes is $O(\log n)$.

Solution:
For an n node tree, its maximum height H can be derived by the equation:

$$\frac{1}{\sqrt{5}}[(\frac{1+\sqrt{5}}{2})^{H+3} - (\frac{1-\sqrt{5}}{2})^{H+3}] - 1 = n$$
$$(\frac{1+\sqrt{5}}{2})^{H+3} - (\frac{1-\sqrt{5}}{2})^{H+3} = \sqrt{5}(n+1)(1')$$

Then

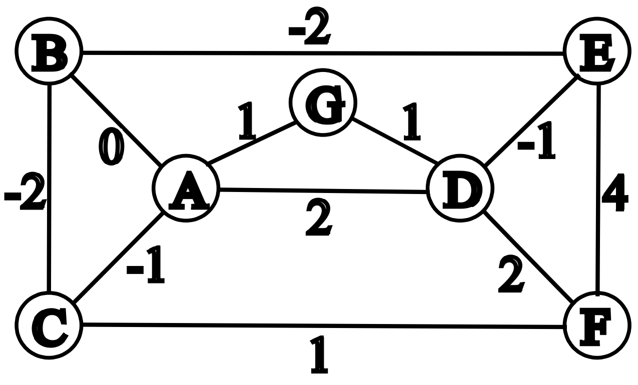
$$(\frac{1+\sqrt{5}}{2})^{H+3} - 1 \leq \frac{1+\sqrt{5}}{2})^{H+3} - (\frac{1-\sqrt{5}}{2})^{H+3} = \sqrt{5}(n+1)(1')$$
$$H \leq \log_{\frac{1+\sqrt{5}}{2}}(\sqrt{5}(n+1)+1) - 3(1')$$
$$H = O(\log n)$$

2.3 Jimmy's Revenge (17 Points)

When Jimmy proudly showed his new system to his superiors, they immediately got angry. One of them told Jimmy: "This is the worst system I have ever seen! If you keep producing such garbage, you will be fired." Jimmy felt very sad, but he did not want to give up. Another superior, who was more sympathetic, told Jimmy: "If you really want to utilize your talents, I have another important task for you. Our fire prevention system is antiquated. If you have the enthusiasm, you can design a new system for us." Jimmy was very excited as he was given a second chance. He immediately started to work on the new system.

i) (3 points) Jimmy's first task is to connect the rooms with pipes. As pipes can be expensive, Jimmy wants to minimize the total length of the pipes, so he decides to find the minimum spanning tree based on the map of the building. However, when he looked at the map, he exclaimed: "Why are there negative weights on the edges?" "Oh, I forgot to mention that," said the superior, "there are some old pipes that are already installed. If you use those lines, you can even recycle some of the pipes." Jimmy was still confused, but as his superior said so, he decided to believe they were really working.

Question: Now, please help Jimmy find the minimum spanning tree with Prim's algorithm based on the following graph. The numbers on the edges are the weights of the edges. Please write down **the order in which the vertices are added to the tree**, and draw **the final MST**. The first node is A.

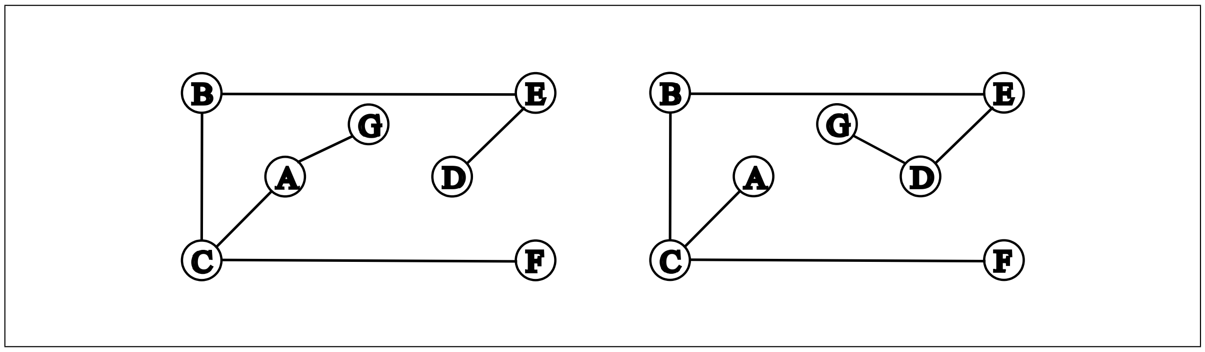


Solution:

One of the following orders is correct:

- $A \to C \to B \to E \to D \to F \to G$
- $A \to C \to B \to E \to D \to G \to F$

One of the following MSTs is correct:



ii) (8 points) When Jimmy looked at another part of the map, he found that there were already some pipes connecting the rooms. He was also informed that they could not buy any new pipes, so he had to use the existing pipes to connect all the rooms. Luckily, the superior told Jimmy that the lengths of the pipes could be ignored. Therefore, if Jimmy wanted to build a new pipe between two rooms, he needed to remove one of the existing pipes. Now the superior wanted Jimmy to find the minimum number of pipes that needed to be removed and rebuilt to connect all the rooms. Jimmy quickly thought of using a Depth-first Search to find all the connected components in the room graph before connecting them together.

Question: Given a graph with n vertices and m edges, an operation is defined as removing one edge and adding another edge at the same time. Please complete the following code to find the minimum number of operations needed to convert the graph into a connected graph. If it is impossible to convert the graph into a connected graph, return -1. Please note that every blank should be filled with one or two lines of code.

Hint: You can use C++-style pseudo code for STL library functions. `vector<int>` is a variable-length array that stores integers, and `vector<vector<int>>` is a 2D array that stores integers and the size of each row can be different.

```
vector<int> visited;
vector<vector<int>> adj_list;

void dfs(int u) {
    ________(1)________
    for (int v : adj_list[u]) {
        // v is the neighbor of u
        ________(2)________
    }
}

int convertToConnectedGraph(int n, vector<vector<int>> connections) {
    if (________(3)________) {
        return -1;
    }

    adj_list.resize(n); // Set the first dimension of the adjacency list to n
    visited.resize(n); // Set the size of the visited array to n
    for (int i = 0; i < n; i++) { // Initialization
        visited[i] = 0;
        adj_list[i] = vector<int>();
    }

    for (auto &edge : connections) {
        int u = edge[0], v = edge[1];
        adj_list[u].push_back(v);
        adj_list[v].push_back(u);
    }

    int ans = 0;
    for (int i = 0; i < n; ++i) {
        if (!visited[i]) {
            ________(4)________
        }
    }

    return ans - 1;
}

```

Solution:

(1) `visited[u] = 1;`

(2) `if (!visited[v]) dfs(v);`

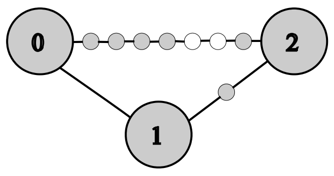
(3) `connections.size() < n - 1`

(4) `dfs(i); ans++;`

iii) (6 points) **(Hard!)** After a long time of hard work, Jimmy finally connected all the rooms. Besides, he also installed some sprinklers, one in each room and one per meter on the pipes. However, when he wanted to test the system, he found that the centralized pump had only a limited water pressure. That is, only the sprinklers within a certain distance from the pump could work. Now Jimmy wanted to find the maximum number of sprinklers that could work. After some consideration, he decided to apply a modified Dijkstra's algorithm to solve this problem.

Question: Given an array of pipes, where each pipe is represented by a vector with three integers $[u, v, s]$, which means there is a pipe between vertex u and vertex v with s sprinklers on it. This also means that the distance between u and v is $s + 1$. Given n vertices including the pump at vertex 0, and `max_dist` representing the maximum distance that the pump can reach, please complete the following code to find the maximum number of sprinklers that can work. Each blank should be filled with one or two lines of code.

Example:



For the graph above, the working sprinklers are marked gray. Given the pipes [[0, 1, 0], [1, 2, 1], [0, 2, 7]] and `max_dist` = 4, the maximum number of working sprinklers is 9.

Hint: You can use C++ style pseudo code for STL library functions.

```cpp
struct Edge {
    int to_v;
    int sprinklers;
    bool operator> (const Edge& that) const {
        if (this->sprinklers != that.sprinklers) return this->sprinklers > that
        .sprinklers;
        else return this->to_v > that.to_v;
    }
};
int reachableSprinkles(vector<vector<int>>& pipes, int max_dist, int n) {
    vector<vector<Edge>> adj_list(n);
    for (auto& pipe : pipes) {
        int u = pipe[0], v = pipe[1], sprinklers = pipe[2];
        adj_list[u].push_back(v,sprinklers);
        adj_list[v].push_back(u,sprinklers);
    }

    // reachable is a n * n matrix
    vector<vector<int>> reachable(n, vector<int>(n));
    vector<int> visited(n);
    int total = 0;

    // pq is a priority queue with pq.top() as the edge with the SMALLEST
    sprinklers
    priority_queue<Edge, vector<Edge>, greater<Edge>> pq;
    pq.push({0,0}); //{to_v, sprinklers}
    while(________(1)________) {
        Edge min_edge = pq.top();
        int u = min_edge.to_v;
        pq.pop();
        if (visited[u]) {
            continue;
        }
        visited[u] = 1;
        total++;
        for (Edge edge: adj_list[u]) {
            int v = edge.to_v;
            int sprinklers = ________(2)________;
            if (________(3)________) {
                pq.push({v, sprinklers});
            }
            reachable[u][v] = ________(4)________;
        }
    }
    for (auto& pipe : pipes) {
        int u = pipe[0], v = pipe[1], sprinklers = pipe[2];
        total += ________(5)________;
    }

    return total;
}
```

Solution:

(1) `!pq.empty() && pq.top().sprinklers <= max_dist`

(2) `edge.sprinklers + min_edge.sprinklers + 1`

(3) `visited[v] == 0`

(4) `min(edge.sprinklers, max_dist - min_edge.sprinklers)`

(5) `min(sprinklers, reachable[u][v] + reachable[v][u])`

2.4 Jimmy's Escape (15 points)

After Jimmy had fully implemented the fire prevention system, he was so tired that he fell asleep in the basement of the building. When he woke up, he was astonished to find the basement flooded! "There must be some problems with the pipes," he thought, but there was no time to hesitate—he had to get out of there! He followed the emergency escape directions and ran. As he was running, he noticed some valuables in the rooms and decided to enter a few of them to grab what he could.

Suppose there are N rooms numbered from 0 to $N-1$, and the room numbered i contains valuables with value v_i. Jimmy was passing the rooms in ascending order. He could enter a room when he passed it and save a value of v_i but he had to give up the next k room. He wanted to maximize the total value he could save from the flooded basement.

You are given an integer array `v[]` representing the value of things in each room, a natural number `N` representing the total number of rooms, and a non-negative integer `k` representing the minimum interval of entering rooms.

Return the maximum value of the things Jimmy could take with him. Suppose all v_i s are no less than 0.

Example 1:

```
Input: v = [3, 1, 6, 10], k = 1
Output: 13
Explanation: 13 = 3 + 10
```

Example 2:

```
Input: v = [4, 2, 12, 0, 7], k = 2
Output: 12
```

Example 3:

```
Input: v = [1, 2, 3], k = 0
Output: 6
```

i) (5 points) A simple solution is to use the dynamic programming method and consider the best he can get if he enters the current (i-th) room, denoted as `dp[i]`. A helper array `currmax` is used to store the maximum of `dp[0]` to `dp[i]`.

```
int max_value(vector<int> &v, int N, int k) {
    vector<int> dp(N, 0), currmax(N, 0);
    dp[0] = v[0];
    currmax[0] = v[0];
    for (int i = 1; i < N; i++) {
        if (i < k) {
            dp[i] = max(dp[i - 1], v[i]);
        } else {
            dp[i] = ________1________;
        }
        currmax[i] = ________2________;
    }
    return ________3________;
}
```

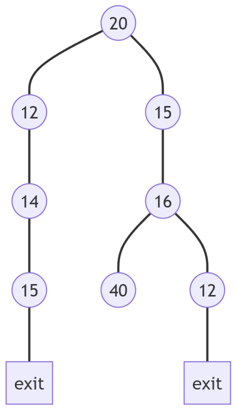
ii) (4 points) Actually, there is no need to define two extra arrays. One `currmax[]` is enough:

```
int max_value2(vector<int> &v, int N, int k) {
    vector<int> currmax(N, 0);
    currmax[0] = v[0];
    for (int i = 1; i < N; i++) {
        if (i < k) {
            currmax[i] = ________4________;
        } else {
            currmax[i] = ________5________;
        }
    }
    return ________3________; // same as max_value()
}
```

iii) (6 points) When Jimmy was escaping according to the escape direction, he found that there were some branch roads he could choose from. He wanted to find a route out where he could get the most value from the rooms.

Suppose each room could have multiple "next room"s, and Jimmy can choose one to continue. Some rooms might have their "next room" recorded as `nullptr`, meaning that after these rooms there are exits. The map guarantees that there are no loops in the rooms and each room cannot be the next room of more than one room.

Example:



```
Input: k = 1, next room structure as above
Output: 36
Explanation: 36 = 20 + 16. Note that there is no exit after 40.
```

```
struct Node {
    int value;
    vector<Node *> next; // Empty if this room is a dead end;
                         // includes a nullptr if there is an exit.
    Node *prev;          // the previous room
    Node *prevk1;        // Preprocessed to store the previous (k+1)th room,
                         // i.e. the closest room that can be chosen
                         // if this room is chosen.

    int max_value;       // the maximum value when Jimmy reaches this room
    //===================== You ONLY need to modify this single field above

    Node(int value, Node *prev, vector<Node *> next) :
```

```cpp
        value(value), prev(prev), next(next), prevk1(nullptr), max_value(0) {}
};

int max_value3(Node *curr, int k, int depth) {
    if (curr == nullptr) return 0;

    // update the max_value of the current room
    if (depth == 0) {
        curr->max_value = curr->value;
    } else if (depth <= k) {
        curr->max_value = max(curr->value, curr->prev->max_value);
    } else {
        curr->max_value = ________6________;
    }

    int res = -1; // if this is a dead end, return -1
    for (auto &n : curr->next) {
        // recurse into next rooms and get return values
        if (n == nullptr) {
            res = max(res, curr->max_value);
        } else {
            res = max(res, ________7________);
        }
    }
    return res;
}
```

```cpp
// ==========================================================================
// IMPORTANT: this main function is only here for reference of the usage of the
// Node structure. You don't have to read it to complete this problem.
int partii_main() {
    int N, k;
    cout << "input N and k: ";
    cin >> N >> k;
    vector<Node> rooms(N, Node(0, nullptr, vector<Node *>()));

    // input the rooms
    for (int i = 0; i < N; i++) {
        cout << "ith room value: ";
        cin >> rooms[i].value;

        cout << "next room numbers:";
        int nextcnt;
        cin >> nextcnt;

        cout << "input ith room nexts: (use -1 to represent exit)" << endl;
        for (int j = 0; j < nextcnt; j++) {
            int nextroom;
            cin >> nextroom;
            if (nextroom == -1) {
                rooms[i].next.push_back(nullptr);
            } else {
                rooms[i].next.push_back(&rooms[nextroom]);
                rooms[nextroom].prev = &rooms[i];
            }
        }
    }

    // preprocess the rooms to store the previous (k+1)th room
    for (int i = 0; i < N; i++) {
        Node *curr = &rooms[i];
        for (int j = 0; j <= k && curr != nullptr; j++) {
            curr = curr->prev;
        }
        rooms[i].prevk1 = curr;
    }

    // call the solving function
    cout << "max value: " << max_value3(&rooms[0], k, 0) << endl;
    return 0;
}
```

Solution:

(1) `max(currmax[i - k - 1] + v[i], dp[i - 1])`

(2) `max(currmax[i - 1], dp[i])`

(3) `currmax[N - 1]`

(4) `max(v[i], currmax[i - 1])`

(5) `max(currmax[i - k - 1] + v[i], currmax[i - 1])`

(6) `max(curr->value + curr->prevk1->max_value,`
`curr->prev->max_value)`

(7) `max_value3(n, k, depth + 1)`

Jimmy finally found the exit and escaped from the basement. Although he was frustrated that his efforts in the fire prevention system were in vain, he was relieved to have gotten out safely. He was also pleased that he managed to save the most valuable items from the flood. His superior was angry with the pipe supplier, and the company had to pay a lot of money to fix the pipes. However, his superior was very satisfied with Jimmy's contribution to the company and his quick response to the emergency. As a result, Jimmy was promoted to Chief Technology Officer. He was very proud of himself and determined to make the company a safer place for everyone.

SCRATCH PAGE:

SCRATCH PAGE: