CCC Sprint Preparation Class

Geyang Liu

Agenda

Fundamental Data Structure

Basic Algorithms

Advanced Algorithms

CCC Sample Questions 3

CCC Sample Questions 4

Fundamental Data Structure

- Array
- List/LinkedList
- Stack
- Queue
- Hash Table (Map/ Set)
- Tree
- Heap/Priority Queue
- Graph

Basic Algorithms

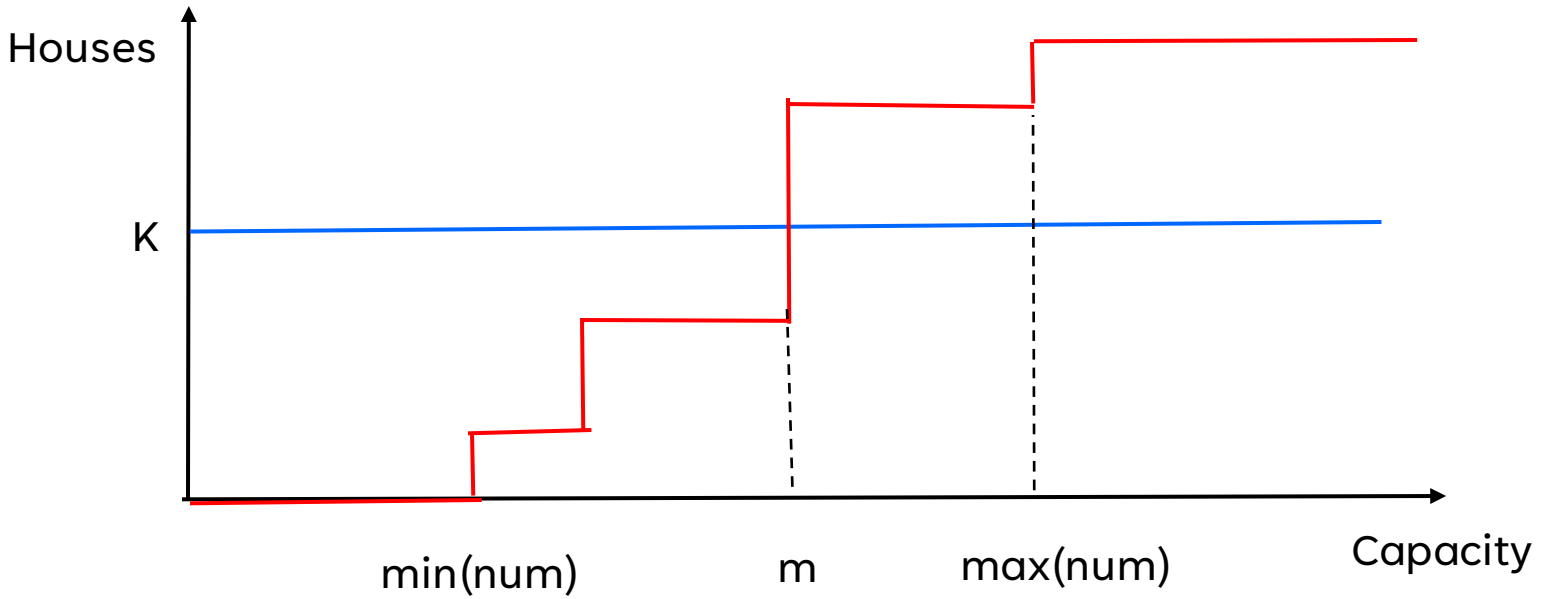
- **Fundamentals (Analysis, Complexity Measures Big(O))**
- Sorting
  - Quick Sort
  - Merge Sort
  - Heap Sort
  - ….
- **Search**
  - Hashing
  - Search trees

Binary Search

LeetCode 2560. House Robber IV https://leetcode.com/problems/house-robber-iv/

1. # of houses we can rob monotonically increases with capacity.

2. Use binary search to find the minimum m so that K <= # of houses.

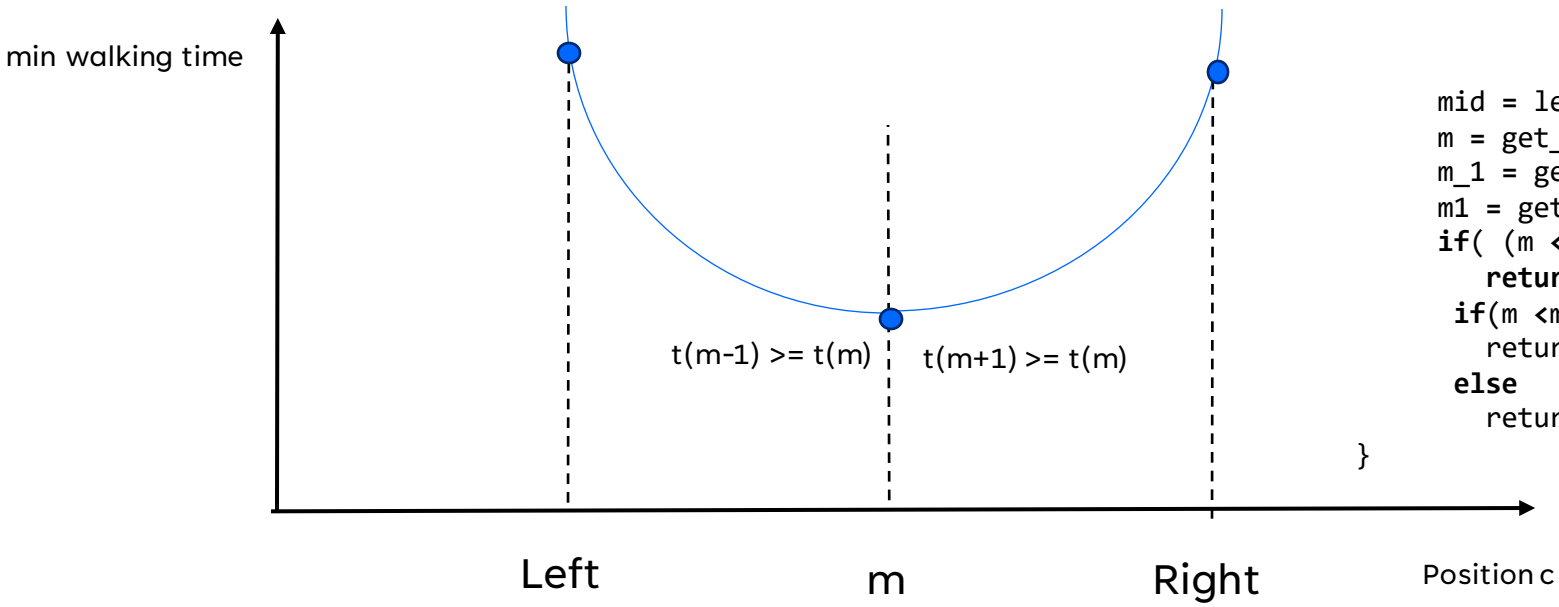



```
class Solution {
    //use binary search to enumerate the stealing ability of the thief.
    public int minCapability(int[] nums, int k) {
      int left = 0, right = (int) 1e9;
      while (left < right) {
        int mid = (left + right) >> 1;
        if (rob(nums, mid) >= k)  right = mid;
        else  left = mid + 1;
      }
      return left;
    }
    // use a greedy approach to determine whether the thief can steal at least x houses.
    private int rob(int[] nums, int x) {
        int cnt = 0, j = -2;
        for (int i = 0; i < nums.length; ++i) {
            if (nums[i] > x || i == j + 1) {
                continue;
            }
            ++cnt;
            j = i;
        }
        return cnt;
    }
}
```

Binary Search

CCC '21 S3 - Lunch Concert

https://dmoj.ca/problem/ccc21s3

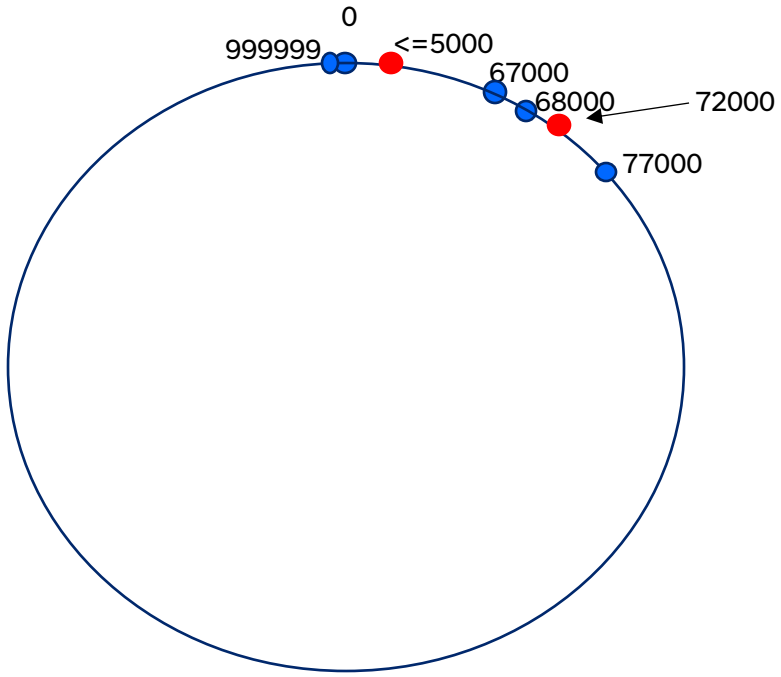



```
get_total_time( position, P, W D)
binary_search(left, right, P, W, D){
  if (left== right)
     return get_total_time(left, P, W,D)
  If(right-left == 1)
     return min(get_total_time(right, P, W,D), get_total_time(left, P, W,D)
   if(right -left == 2)
       return min(get_total_time(right,P,W,D),
                  get_total_time(left,P,W,D));
                  getSum(left + 1,P,W,D));

      mid = left + (right-left)/2;
      m = get_total_time(mid,P,W,D);
      m_1 = get_total_time(mid -1,P,W,D);  // m-1
      m1 = get_total_time(mid + 1,P,W,D);  // m+1
      if( (m <= m_1) && (m<=m1))// t(m)<= t(m-1) && t(m)<=t(m_+1)
         return m;
       if(m <m_1)
         return binary_search(mid, right, P,W,D);
       else
         return binary_search(left, mid, P,W,D);
}
```

Binary Search

CCC '10 S3 - Firehose




```
binary_search(H, K){
   houses = new int[H*2];
   // scan H houses to houses
   sort(houses);
   for (i = 0; i < H; i++)
      houses[i + H] = houses[i] + 1M;
   lo = 0; hi = 1M;
   while (lo < hi) {
     int mid = lo + (hi - lo) / 2;
     if (getMinHoseCount(houses,H, mid) > K) lo = mid + 1;
     else hi = mid;
   }
   print(lo);
}
// return mininum hose count needed
int getMinHoseCount(houses, H,len) {
     min = MAX_VALUE;
     for (i = 0; i < H; i++) {
       cur = i;h = 1;
       for (j = i; j < H + i; j++)
         if (houses[j] - houses[cur] > 2 * len) {cur = j;
                                                 h++;}
       min = Math.min(min, h);
     }
     return min;
             }
```

Graph Problems

- Adjacency Matrix and Adjacency List
- BFS & DFS in Graph
- Cycles in Graph
- Shortest Paths in Graph
- Minimum Spanning Tree
- Topological Sorting

BFS Breadth-First Search

- Start at the root of the graph and visits all nodes at the current depth level before moving on to the nodes at the next depth level.
- 1. start at node
- 2. Visit all the nodes' neighbors
- 3. Visit all the neighbors' neighbors
- 4. continues in this fashion.
- https://www.cs.usfca.edu/~galles/visualization/BFS.html
- https://github.com/rayliu7717/CCC_CLASS/blob/main/CCC_GRAPH_BFS.java

BFS Problems

- Shortest Path and Minimum Spanning Tree
- Level Order traverse tree
- Cycle detection in graph
- Path Finding
- Finding all nodes within one connected component
- Connected Component
- Topological sorting
- .....

BFS Sample Problems

- Leetcode 111. Minimum Depth of Binary Tree
- Leetcode 102. Binary Tree Level Order Traversal
- Leetcode 127, Word Ladder

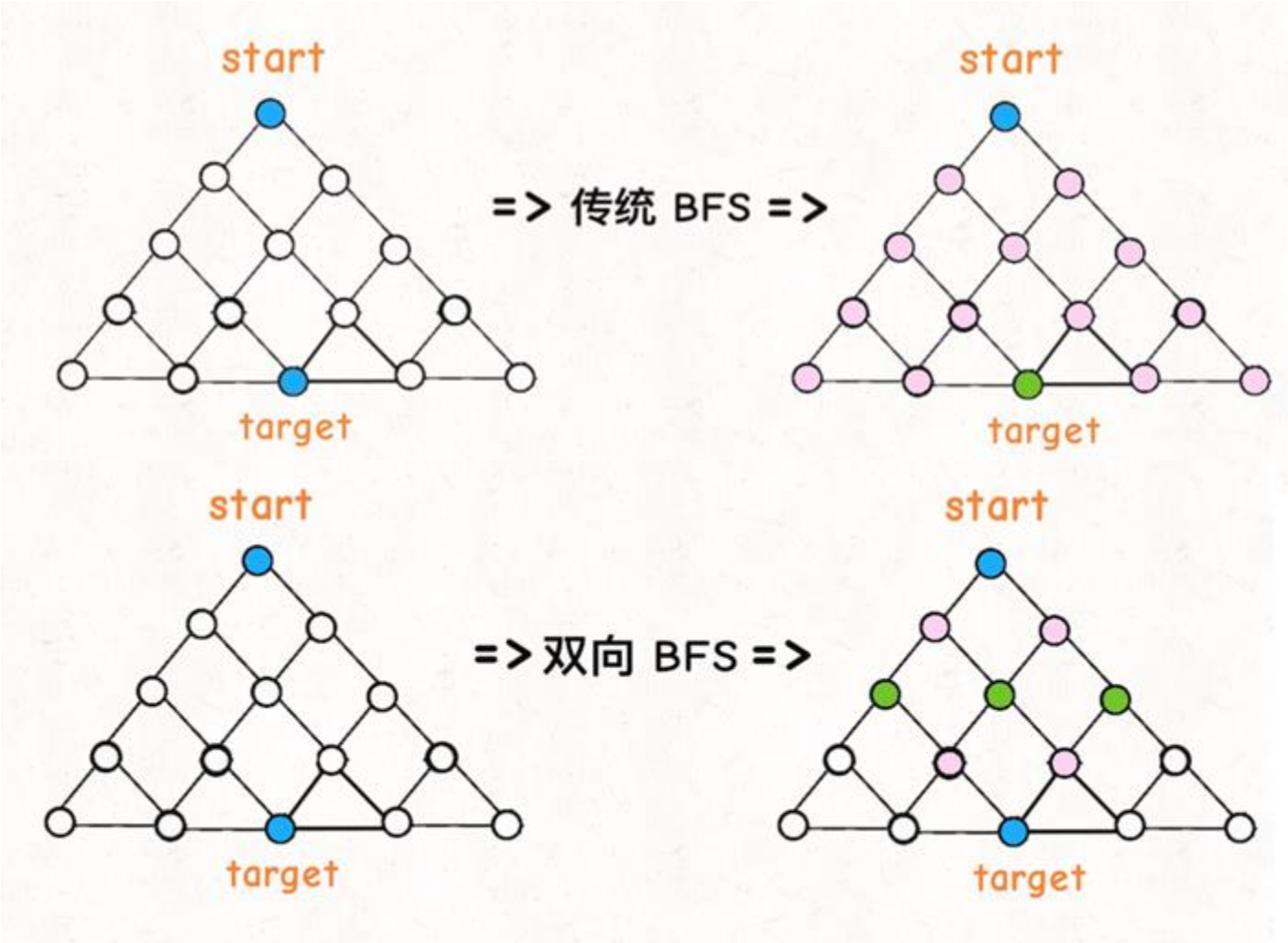
  https://leetcode.com/problems/word-ladder/submissions/
- Leetcode 207, Course Schedule

  https://leetcode.com/problems/course-schedule/

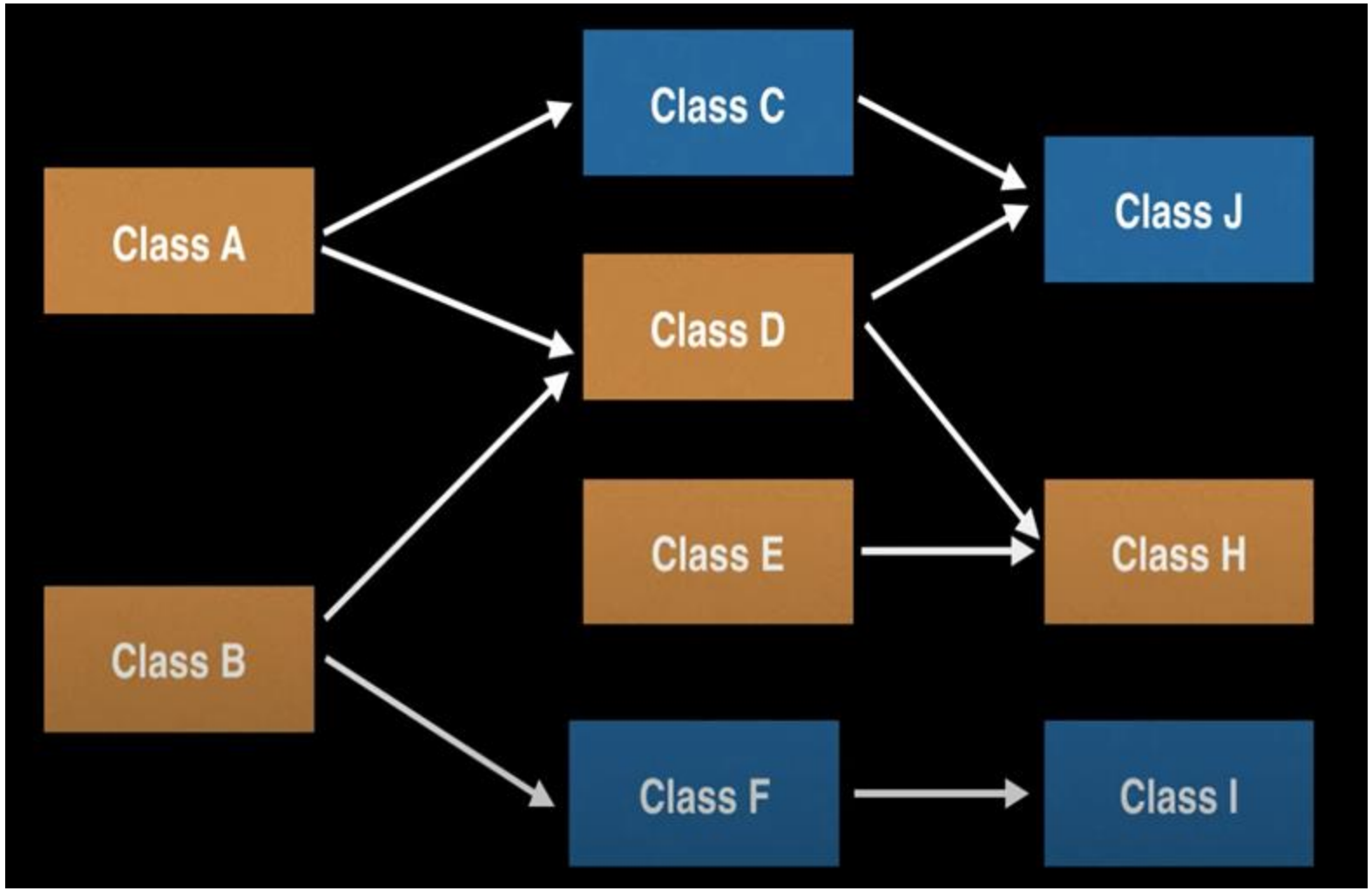
  https://www.cs.usfca.edu/~galles/visualization/TopoSortIndegree.html

Optimize BFS
(Leetcode 127, Word Ladder ，Leetcode 752 Open the Lock)

One way direction BFS searches from the start node expanding down until gets to the end node. While bi way direction BFS searches from both start node and end nodes and expand from both sides, until two search node sets have the intersections 。



BFS in Topological Sort (Leetcode 207, Course Schedule)




```
Require: G is a directed acyclic graph (DAG)
 1: function TopSort(G)
 2:     T ← empty list
 3:     Z ← empty queue/stack/whatever
 4:     in ← dictionary mapping all vertices to 0
 5:     for each v ∈ V do
 6:         for each u adjacent to v do
 7:             increment in[v]
 8:     for each v ∈ V do
 9:         if in[v] = 0 then
10:             add v to Z
11:     while S is not empty do
12:         v ← Z.remove
13:         append v to T
14:         for each u adjacent to v do
15:             decrement in[u]
16:             if in[u] = 0 then
17:                 add u to Z
        return T
```

DFS Depth-First Search

- Start at the root node and explore as far as possible along each branch before backtracking.
- Graphs may contain cycles (a node may be visited twice). To avoid processing a node more than once, use a boolean visited array. A graph can have more than one DFS traversal.
- https://www.cs.usfca.edu/~galles/visualization/DFS.html
- https://github.com/rayliu7717/CCC_CLASS/blob/main/CCC_GRAPH_DFS.java

DFS Template

```
Result = []
void DFS ( path, list)
    if(match exist condition)
            result.add(path)
    For (item : list)
        select this item
        DFS ( path, list) // back track
        cancel the selection
```

DFS is a Brute Force to enumerate all combinations.
Enumerate recursively,
Cannot use "for loop" to implement since we don't know how many loop level yet.

DFS Classic Problems

- Leetcode 78 Subset

  https://leetcode.com/problems/subsets/
- Leetcode 46 Permutations
- https://leetcode.com/problems/permutations/submissions/
- Leetcode 77 Combinations
- Leetcode 37 Sudoku Solver
- Leetcode 51 N-Queens

Travel Plan (DFS)

There are n cities, and the adjacency matrix `arr` represents the distance between any two cities.`arr[i][j]` represents the distance from city i to city j .Alice made a travel plan on the weekend. She started from city 0, then she traveled other cities 1 ~ n-1, and finally returned to city 0. Alice wants to know the minimum distance she needs to walk to complete the travel plan. Return this minimum distance. Except for city 0, every city can only pass once, and city 0 can only be the starting point and destination. Alice can't pass city 0 during travel.

Input:

[[0,1,2],[1,0,2],[2,1,0]]

Output:

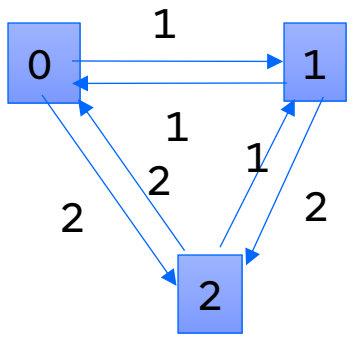
4

Explanation:

There are two possible plans.

The first, city 0-> city 1-> city 2-> city 0, cost = 5.

The second, city 0-> city 2-> city 1-> city 0, cost = 4.




```java
public class Solution {
    /**
     * @param arr: the distance between any two cities
     * @return: the minimum distance Alice needs to walk to
complete the travel plan
     */
    void dfs(int [][] arr, int nowpos, int n, boolean[] vis, int
sum, int cnt, int[] ans )
    {
        // exit
        if(cnt == n -1){
            ans[0] = Math.min(ans[0], sum+ arr[nowpos][0]);
            return;
        }
        for(int i = 1;i<n; ++i){
            if(!vis[i]){
                vis[i] = true;
                dfs(arr,i, n, vis,sum+ arr[nowpos][i], cnt + 1,
ans);
                vis[i] = false;  //backtrak
            }
        }
    }
    public int travelPlan(int[][] arr) {
        // Write your code here.
        int n = arr.length;
        boolean [] vis = new boolean[n];
        int[] ans = new int[1];
        ans[0] = Integer.MAX_VALUE;
        dfs(arr, 0, n, vis, 0, 0,ans);
        return ans[0];
    }
}
```

CCC '04 S3 – Spreadsheet (DFS)

1. Each Cell is a Graph Node (id is r,c)
2. Letter with number is graph edge
3. Traverse each Node to check circle
4. All the nodes on the circle will mark with "*"
5. If no circle, calculate the sum number.

```
int dfs(int r, int c, grid[][], vis[][]) {
  if (isNumeric(grid[r][c])) return grid[r][c].toInt();
  if (vis[r][c] || grid[r][c] == "*") return -1;
  vis[r][c] = true;
  String [] depend = grid[r][c].split("+");
   int sum = 0;
   for (int i = 0; i < depend.length; i++) {
      int ret = dfs(depend[i].charAt(0)-'A', depend[i].charAt(1)-'0' - 1);
      if (ret==-1) { grid[r][c] = "*"; return -1;}
      else sum+=ret;
   }
   grid[r][c] = sum.toString();
   return sum;
}

void spreadsheet(grid, rows, cols)
{
    for (int i = 0; i < rows; i++) {
       for (int j = 0; j < cols; j++) {
          vis = new boolean[rows][cols];
           dfs(i, j, grid, vis );
       }
    }
}
```

Greedy Algorithm

- Build up a solution piece by piece, always choosing the next piece that offers the most obvious and immediate benefit.
- The problems where choosing locally optimal also leads to global solution are the best fit for Greedy.

Thank you

Gerry Liu

Liu.geyang@gmail.com