

# t0626

---

华为od 机试 2025B卷 - 乘坐保密电梯 (C++ & Python & JAVA & JS

华为OD机试 2025 B卷 - 伐木工 (C++ & Python & JAVA & JS & G

华为OD机试 2025 B卷 - 查找接口成功率最优时间段 (C++ & Python & JAVA

华为OD机试 2025 B卷 - 抢7游戏 (C++ & Python & JAVA & JS &

华为OD 机试 2025 B卷 - 数组二叉树 (C++ & Python & JAVA & JS

华为OD 机试 2025 B卷 - 路灯照明问题 (C++ & Python & JAVA & JS

华为OD机试 2025 B卷 - 最佳投资方式 (C++ & Python & JAVA & JS

华为OD 机考 2025B卷 - 计算网络信号 / 信号强度 (C++ & Python & JAV

# 华为od 机试 2025B卷 - 乘坐保密电梯 (C++ & Python & JAVA & JS)

## 乘坐保密电梯

2025B卷目录点击查看: [华为OD机试2025B卷真题题库目录](#) | [机考题库](#) + [算法考点详解](#)

2025B卷 100分题型

## 题目描述

有一座保密大楼，你从0楼到达指定楼层m，必须这样的规则乘坐电梯：

给定一个数字序列，每次根据序列中的数字n，上升n层或者下降n层，前后两次的方向必须相反，规定首次的方向向上，自行组织序列的顺序按规定操作到达指定楼层。

求解到达楼层的序列组合，如果不能到达楼层，给出小于该楼层的最近序列组合。

## 输入描述

第一行：期望的楼层，取值范围[1,50]; 序列总个数，取值范围[1,23]

第二行：序列，每个值取值范围[1,50]

## 输出描述

能够达到楼层或者小于该楼层最近的序列

## 备注

- 操作电梯时不限定楼层范围。
- 必须对序列中的每个项进行操作，不能只使用一部分。

## 用例1

### 输入

### 输出

## 说明

1 2 6, 6 2 1均为可行解，按先处理大值的原则结果为6 2 1

## 题解

思路：递归回溯

- 首先确定需要选取上升的数量,规定第一个为上升并且交叉上升和下降,那么选取上升的数量为  $(n + 1) / 2$ ,向上取整。
- 将输入数组降序排序。
- 使用 递归回溯 枚举所有选取 `exceptCount` 上升数,记录其中出现 上升高度  $\leq m$ , 并且最接近  $m$  的方案。
- 将每个候选方案对应的数组构建出来,选取其中字典序最大的数组就是结果。

C++

```
1  #include<iostream>
2  #include<vector>
3  #include<string>
4  #include <utility>
5  #include <sstream>
6  #include<algorithm>
7  #include<list>
8  #include<queue>
9  #include<map>
10 #include<set>
11 using namespace std;
12
13 bool cmp(int x, int y) {
14     return x > y;
15 }
16 // 整体序列和
17 int totalSum;
18 vector<int> res;
19 int minDiff;
20 int m,n;
21
22 // 递归回溯枚举 选取exceptCount的数
23 void DFS(vector<int>& ans, int index , int currentSum, int count, int exceptCount, int& visited) {
24     if (count > exceptCount) {
25         return;
26     }
27     if (exceptCount == count) {
28         // 净增楼层
29         int diff = currentSum - (totalSum - currentSum);
30         // 上升超过m的不考虑
31         if (diff > m) {
32             return;
33         }
34
35         // 找最小差距
36         diff = abs(diff - m);
37         if (diff < minDiff) {
38             res.clear();
39             minDiff = diff;
40             res.push_back(visited);
41         } else if (diff == minDiff) {
42             res.push_back(visited);
43         }
44         return;
45     }
```

```

45     }
46
47     for (int i = index; i < n; i++) {
48         int num = ans[i];
49         int nextCurrentSum = currentSum + num;
50         // 剪枝 最终会超过m层
51         if (nextCurrentSum - (totalSum - nextCurrentSum) > m) {
52             continue;
53         }
54         // 递归回溯
55         visited |= 1 << (n - i - 1);
56         DFS(ans, i + 1, nextCurrentSum, count + 1, exceptCount, visited);
57         visited &= ~(1 << (n - i - 1));
58
59     }
60 }
61
62 // 交叉构建结果集
63 vector<int> buildArr(vector<int>& ans, int num) {
64     vector<bool> visted(n, false);
65     for (int i = 0; i < n; i++) {
66         visted[i] = 1 & (num >> (n-1-i));
67     }
68     vector<int> s(n);
69     int pos = 0;
70     for (int i = 0; i < n; i++) {
71         if (visted[i]) {
72             s[pos] = ans[i];
73             pos += 2;
74         }
75     }
76     pos = 1;
77     for (int i = 0; i < n; i++) {
78         if (!visted[i]) {
79             s[pos] = ans[i];
80             pos += 2;
81         }
82     }
83     return s;
84 }
85
86 // 在所有mask生成的数组中，找出字典序最大的
87 vector<int> findBestArray(vector<int>& ans) {
88     vector<int> best;
89
90     for (const int& mask : res) {
91         vector<int> curr = buildArr(ans, mask);
92         if (best.empty() || curr > best) {

```

```

93         best = curr;
94     }
95 }
96 return best;
97 }
98
99 int main() {
100     cin >> m >> n;
101     vector<int> ans(n);
102     int sum = 0;
103     for (int i = 0; i < n; i++) {
104         cin >> ans[i];
105         sum += ans[i];
106     }
107     //
108     totalSum = sum;
109     minDiff = sum + m;
110     // 从大到小排序
111     sort(ans.begin(), ans.end(), cmp);
112     // 向上取整 向上的数量
113     int exceptUpCount = (n + 1) / 2;
114
115     int status = 0;
116     DFS(ans, 0, 0, 0, exceptUpCount, status);
117
118     vector<int> answer = findBestArray(ans);
119     for (int i = 0; i < n; i++) {
120         cout << answer[i];
121         if (i != n-1) {
122             cout << " ";
123         }
124     }
125     return 0;
126 }

```

## JAVA

```
1  import java.util.*;
2
3  public class Main {
4      static int totalSum;
5      static List<Integer> res = new ArrayList<>();
6      static int minDiff;
7      static int m, n;
8
9      // 比较函数, 从大到小排序
10     static boolean cmp(int x, int y) {
11         return x > y;
12     }
13
14     // 递归回溯枚举 选取exceptCount的数
15     static void DFS(List<Integer> ans, int index, int currentSum, int count, int exceptCount, int visited) {
16         if (count > exceptCount) {
17             return;
18         }
19         if (exceptCount == count) {
20             // 净增楼层
21             int diff = currentSum - (totalSum - currentSum);
22             // 上升超过m的不考虑
23             if (diff > m) {
24                 return;
25             }
26             // 找最小差距
27             diff = Math.abs(diff - m);
28             if (diff < minDiff) {
29                 res.clear();
30                 minDiff = diff;
31                 res.add(visited);
32             } else if (diff == minDiff) {
33                 res.add(visited);
34             }
35             return;
36         }
37
38         for (int i = index; i < n; i++) {
39             int num = ans.get(i);
40             int nextCurrentSum = currentSum + num;
41             // 剪枝 最终会超过m层
42             if (nextCurrentSum - (totalSum - nextCurrentSum) > m) {
43                 continue;
44             }
```

```

45         // 递归回溯
46         visited |= 1 << (n - i - 1);
47         DFS(ans, i + 1, nextCurrentSum, count + 1, exceptCount, visit
ed);
48         visited &= ~(1 << (n - i - 1));
49     }
50 }
51
52 // 交叉构建结果集
53 static List<Integer> buildArr(List<Integer> ans, int num) {
54     boolean[] visited = new boolean[n];
55     for (int i = 0; i < n; i++) {
56         visited[i] = (1 & (num >> (n - 1 - i))) == 1;
57     }
58     List<Integer> s = new ArrayList<>(Collections.nCopies(n, 0));
59     int pos = 0;
60     for (int i = 0; i < n; i++) {
61         if (visited[i]) {
62             s.set(pos, ans.get(i));
63             pos += 2;
64         }
65     }
66     pos = 1;
67     for (int i = 0; i < n; i++) {
68         if (!visited[i]) {
69             s.set(pos, ans.get(i));
70             pos += 2;
71         }
72     }
73     return s;
74 }
75
76 // 在所有mask生成的数组中，找出字典序最大的
77 static List<Integer> findBestArray(List<Integer> ans) {
78     List<Integer> best = new ArrayList<>();
79     for (int mask : res) {
80         List<Integer> curr = buildArr(ans, mask);
81         if (best.isEmpty() || compare(curr, best) > 0) {
82             best = curr;
83         }
84     }
85     return best;
86 }
87
88 // 手动比较两个列表的字典序
89 static int compare(List<Integer> a, List<Integer> b) {
90     for (int i = 0; i < a.size(); i++) {
91         if (!a.get(i).equals(b.get(i))) {

```



```

92         return a.get(i) - b.get(i);
93     }
94 }
95     return 0;
96 }
97
98 public static void main(String[] args) {
99     Scanner sc = new Scanner(System.in);
100     m = sc.nextInt();
101     n = sc.nextInt();
102     List<Integer> ans = new ArrayList<>();
103     totalSum = 0;
104     for (int i = 0; i < n; i++) {
105         int x = sc.nextInt();
106         ans.add(x);
107         totalSum += x;
108     }
109     minDiff = totalSum + m;
110     // 从大到小排序
111     ans.sort((a, b) -> b - a);
112
113     int exceptUpCount = (n + 1) / 2;
114     int status = 0;
115     DFS(ans, 0, 0, 0, exceptUpCount, status);
116
117     List<Integer> answer = findBestArray(ans);
118     for (int i = 0; i < answer.size(); i++) {
119         System.out.print(answer.get(i));
120         if (i != answer.size() - 1) {
121             System.out.print(" ");
122         }
123     }
124 }
125 }

```

## Python

```
1 totalSum = 0
2
3 res = []
4
5 minDiff = 0
6 m, n = 0, 0
7
8 def DFS(ans, index, currentSum, count, exceptCount, visited):
9     global minDiff, res, totalSum, m, n
10    if count > exceptCount:
11        return
12    if count == exceptCount:
13
14        diff = currentSum - (totalSum - currentSum)
15
16        if diff > m:
17            return
18
19        diff = abs(diff - m)
20        if diff < minDiff:
21            res.clear()
22            minDiff = diff
23            res.append(visited)
24        elif diff == minDiff:
25            res.append(visited)
26        return
27
28    for i in range(index, n):
29        num = ans[i]
30        nextCurrentSum = currentSum + num
31
32        if nextCurrentSum - (totalSum - nextCurrentSum) > m:
33            continue
34
35        visited |= 1 << (n - i - 1)
36        DFS(ans, i + 1, nextCurrentSum, count + 1, exceptCount, visited)
37        visited &= ~(1 << (n - i - 1))
38
39
40 def buildArr(ans, mask):
41     visited = [(mask >> (n - 1 - i)) & 1 for i in range(n)]
42     s = [0] * n
43     pos = 0
44     for i in range(n):
45         if visited[i]:
```

```

46         s[pos] = ans[i]
47         pos += 2
48     pos = 1
49     for i in range(n):
50         if not visited[i]:
51             s[pos] = ans[i]
52             pos += 2
53     return s
54
55
56 def findBestArray(ans):
57     best = []
58     for mask in res:
59         curr = buildArr(ans, mask)
60         if not best or curr > best:
61             best = curr
62     return best
63
64 if __name__ == "__main__":
65     m, n = map(int, input().split())
66     ans = list(map(int, input().split()))
67     totalSum = sum(ans)
68     minDiff = totalSum + m
69
70     ans.sort(reverse=True)
71
72     exceptUpCount = (n + 1) // 2
73     status = 0
74     DFS(ans, 0, 0, 0, exceptUpCount, status)
75     answer = findBestArray(ans)
76     print(*answer)

```

## JavaScript

```
1  const readline = require('readline');
2
3  const rl = readline.createInterface({
4    input: process.stdin,
5    output: process.stdout
6  });
7
8  let input = [];
9  rl.on('line', function (line) {
10    input.push(...line.trim().split(' ').map(Number));
11  }).on('close', function () {
12    let idx = 0;
13    const m = input[idx++];
14    const n = input[idx++];
15    const ans = [];
16    for (let i = 0; i < n; i++) {
17      ans.push(input[idx++]);
18    }
19    solve(m, n, ans);
20  });
21
22  let totalSum;
23  let res = [];
24  let minDiff;
25  let m_, n_;
26
27  function solve(m, n, ans) {
28    m_ = m;
29    n_ = n;
30    totalSum = ans.reduce((a, b) => a + b, 0);
31    minDiff = totalSum + m;
32
33    ans.sort((a, b) => b - a);
34
35    let exceptUpCount = Math.floor((n + 1) / 2);
36    let status = 0;
37    DFS(ans, 0, 0, 0, exceptUpCount, status);
38
39    const answer = findBestArray(ans);
40    console.log(answer.join(' '));
41  }
42
43
44  function DFS(ans, index, currentSum, count, exceptCount, visited) {
45    if (count > exceptCount) {
```

```

46         return;
47     }
48     if (count === exceptCount) {
49
50         let diff = currentSum - (totalSum - currentSum);
51
52         if (diff > m_) {
53             return;
54         }
55
56         diff = Math.abs(diff - m_);
57         if (diff < minDiff) {
58             res = [];
59             minDiff = diff;
60             res.push(visited);
61         } else if (diff === minDiff) {
62             res.push(visited);
63         }
64         return;
65     }
66     for (let i = index; i < n_; i++) {
67         let num = ans[i];
68         let nextCurrentSum = currentSum + num;
69
70         if (nextCurrentSum - (totalSum - nextCurrentSum) > m_) {
71             continue;
72         }
73
74         visited |= 1 << (n_ - i - 1);
75         DFS(ans, i + 1, nextCurrentSum, count + 1, exceptCount, visited);
76         visited &= ~(1 << (n_ - i - 1));
77     }
78 }
79
80
81 function buildArr(ans, mask) {
82     let visited = Array.from({length: n_}, (_, i) => (mask >> (n_ - 1 -
83 i)) & 1);
84     let s = Array(n_).fill(0);
85     let pos = 0;
86     for (let i = 0; i < n_; i++) {
87         if (visited[i]) {
88             s[pos] = ans[i];
89             pos += 2;
90         }
91     }
92     pos = 1;
93     for (let i = 0; i < n_; i++) {

```

```

93         if (!visited[i]) {
94             s[pos] = ans[i];
95             pos += 2;
96         }
97     }
98     return s;
99 }
100
101
102 function findBestArray(ans) {
103     let best = [];
104     for (let mask of res) {
105         let curr = buildArr(ans, mask);
106         if (best.length === 0 || compare(curr, best) > 0) {
107             best = curr;
108         }
109     }
110     return best;
111 }
112
113 function compare(a, b) {
114     for (let i = 0; i < a.length; i++) {
115         if (a[i] !== b[i]) {
116             return a[i] - b[i];
117         }
118     }
119     return 0;
120 }

```

**Go**

```
1 package main
2
3 import (
4     "fmt"
5     "sort"
6 )
7
8 var (
9     totalSum int
10    res      []int
11    minDiff  int
12    m, n     int
13 )
14
15 func main() {
16     fmt.Scan(&m, &n)
17     ans := make([]int, n)
18     sum := 0
19     for i := 0; i < n; i++ {
20         fmt.Scan(&ans[i])
21         sum += ans[i]
22     }
23     totalSum = sum
24     minDiff = sum + m
25
26     sort.Slice(ans, func(i, j int) bool {
27         return ans[i] > ans[j]
28     })
29
30     exceptUpCount := (n + 1) / 2
31     status := 0
32     DFS(ans, 0, 0, 0, exceptUpCount, status)
33     answer := findBestArray(ans)
34     for i := 0; i < n; i++ {
35         fmt.Print(answer[i])
36         if i != n-1 {
37             fmt.Print(" ")
38         }
39     }
40     fmt.Println()
41 }
42
43
44 func DFS(ans []int, index, currentSum, count, exceptCount, visited int) {
45     if count > exceptCount {
```

```

46     return
47 }
48 if count == exceptCount {
49
50     diff := currentSum - (totalSum - currentSum)
51
52     if diff > m {
53         return
54     }
55
56     if d := abs(diff - m); d < minDiff {
57         res = []int{}
58         minDiff = d
59         res = append(res, visited)
60     } else if d == minDiff {
61         res = append(res, visited)
62     }
63     return
64 }
65
66 for i := index; i < n; i++ {
67     num := ans[i]
68     nextCurrentSum := currentSum + num
69
70     if nextCurrentSum-(totalSum-nextCurrentSum) > m {
71         continue
72     }
73
74     visited |= 1 << (n - i - 1)
75     DFS(ans, i+1, nextCurrentSum, count+1, exceptCount, visited)
76     visited &= ^(1 << (n - i - 1))
77 }
78 }
79
80
81 func buildArr(ans []int, mask int) []int {
82     visited := make([]bool, n)
83     for i := 0; i < n; i++ {
84         visited[i] = (mask>>(n-1-i))&1 == 1
85     }
86     s := make([]int, n)
87     pos := 0
88     for i := 0; i < n; i++ {
89         if visited[i] {
90             s[pos] = ans[i]
91             pos += 2
92         }
93     }

```



```

94     pos = 1
95     for i := 0; i < n; i++ {
96         if !visited[i] {
97             s[pos] = ans[i]
98             pos += 2
99         }
100     }
101     return s
102 }
103
104
105 func findBestArray(ans []int) []int {
106     var best []int
107     for _, mask := range res {
108         curr := buildArr(ans, mask)
109         if best == nil || compare(curr, best) > 0 {
110             best = curr
111         }
112     }
113     return best
114 }
115
116 func compare(a, b []int) int {
117     for i := 0; i < len(a); i++ {
118         if a[i] != b[i] {
119             return a[i] - b[i]
120         }
121     }
122     return 0
123 }
124
125 func abs(x int) int {
126     if x < 0 {
127         return -x
128     }
129     return x
130 }

```

来自: [华为od 机试 2025B卷 – 乘坐保密电梯 \(C++ & Python & JAVA & JS & GO\)\\_2025华为od机试题-CSDN博客](#)

# 华为OD机试 2025 B卷 - 伐木工 (C++ & Python & JAVA & JS & G)

## 伐木工

华为OD机试真题目录点击查看: [华为OD机试2025B卷真题题库目录](#) | [机考题库](#) + [算法考点详解](#)

华为OD机试2025B卷 100分题型

## 题目描述

一根X米长的树木，伐木工切割成不同长度的木材后进行交易，交易价格为每根木头长度的乘积。规定切割后的每根木头长度都为正整数；也可以不切割，直接拿整根树木进行交易。请问伐木工如何尽量少的切割，才能使收益最大化？

## 输入描述

木材的长度 ( $X \leq 50$ )

## 输出描述

输出最优收益时的各个树木长度，以空格分隔，按[升序排列](#)

## 用例1

### 输入

### 输出

## 说明

一根2米长的树木，伐木工不切割，为 $2 * 1$ ，收益最大为2

一根4米长的树木，伐木工不需要切割为 $2 * 2$ ，省去切割成本，直接整根树木交易，为 $4 * 1$ ，收益最大为4

一根5米长的树木，伐木工切割为 $2 * 3$ ，收益最大为6

一根10米长的树木，伐木工可以切割方式一：3，4，4，也可以切割为方式二：3，2，2，3，但方式二伐木工多切割一次，增加切割成本却买了一样的价格，因此并不是最优收益。

## 题解

思路： 动态规划 + DFS

1. 题目要求最大价值前提下，最小切割率的切割方式。
2. 定义 `dp[]` 数组存储内容格式为 {最大价值, 最小切割数}，其中 `dp[i]` 表示含义为x的长度的木板能够获得最大收益并且最小切割数。为了得出切割方案，定义 `path[]` 数组，其中 `path[i]` 表示能获得最大收益且最小切割数的切割长度。
3. 基本逻辑如下：
  - 对于长度为j的木板，初始设置 `dp[j] = {j, 0}`, `path[j] = 0` .
  - 枚举切割值 `i` ( $1 \leq x < j$ )，如果 `dp[j - i][0] * dp[i][0] >= dp[j][0]` (能获得更大收益)或者 `dp[j-i][0] * dp[i][0] == dp[j][0] and dp[j-i][1] + dp[i][1] + 1 < dp[j][1]` 则更新 `dp[j][0] = dp[i][0] * dp[j - i][0]`, `dp[j][1] = dp[i][1] + dp[j - i][1] + 1`, `path[j] = i`
4. 基于3的规律，枚举j为[1,x]。就能得到长度为x的最大收益以及每一步的切割方案。利用 `path[]` 方程递归就能得到具体的切割路径。
5. 将通过 DFS 得到的切割路径，按照升序排序。输出切割路径。

C++

```
1  #include <iostream>
2  #include <vector>
3  #include <algorithm>
4  using namespace std;
5
6  // 回溯切割路径
7  void backtrack(int n, const vector<int>& path, vector<int>& res) {
8      if (path[n] == 0) {
9          res.push_back(n); // 不切割, 整根加入
10     } else {
11         backtrack(path[n], path, res);
12         backtrack(n - path[n], path, res);
13     }
14 }
15
16 int main() {
17     int X;
18     cin >> X;
19
20     // dp[i] = pair<最大乘积, 最小切割次数>
21     vector<pair<int, int>> dp(X + 1, {0, 0});
22     vector<int> path(X + 1, 0); // path[i] 表示长度为 i 的第一次切割位置
23
24     for (int i = 1; i <= X; ++i) {
25         int maxVal = i;
26         int minCuts = 0;
27         int cutPos = 0;
28
29         for (int j = 1; j < i; ++j) {
30             int leftVal = dp[j].first;
31             int leftCuts = dp[j].second;
32             int rightVal = dp[i - j].first;
33             int rightCuts = dp[i - j].second;
34
35             int prod = leftVal * rightVal;
36             int cuts = leftCuts + rightCuts + 1;
37
38             // 优先选最大收益, 收益相同时选切割更少的方案
39             if (prod > maxVal || (prod == maxVal && cuts < minCuts)) {
40                 maxVal = prod;
41                 minCuts = cuts;
42                 cutPos = j;
43             }
44         }
45     }
```

```

46         dp[i] = {maxVal, minCuts};
47         path[i] = cutPos;
48     }
49
50     vector<int> res;
51     backtrack(X, path, res);
52     sort(res.begin(), res.end());
53
54     for (int i = 0; i < res.size(); ++i) {
55         cout << res[i];
56         if (i < res.size() - 1)
57             cout << " ";
58     }
59     cout << endl;
60
61     return 0;
62 }

```

## JAVA

```
1  import java.util.*;
2
3  public class Main {
4      public static void main(String[] args) {
5          Scanner sc = new Scanner(System.in);
6          int X = sc.nextInt();
7
8          // dp[i] = [最大收益, 最小切割次数]
9          int[][] dp = new int[X + 1][2];
10         int[] path = new int[X + 1]; // path[i] 表示长度为 i 时的第一刀切的位置 (0 表示不切)
11
12         for (int i = 1; i <= X; i++) {
13             int maxVal = i;          // 不切割的收益
14             int minCuts = 0;
15             int cutPos = 0;
16
17             for (int j = 1; j < i; j++) {
18                 int leftVal = dp[j][0], leftCuts = dp[j][1];
19                 int rightVal = dp[i - j][0], rightCuts = dp[i - j][1];
20
21                 int prod = leftVal * rightVal;
22                 int cuts = leftCuts + rightCuts + 1;
23
24                 // 优先收益大, 其次切割次数少
25                 if (prod > maxVal || (prod == maxVal && cuts < minCuts)) {
26                     maxVal = prod;
27                     minCuts = cuts;
28                     cutPos = j;
29                 }
30             }
31
32             dp[i][0] = maxVal;
33             dp[i][1] = minCuts;
34             path[i] = cutPos;
35         }
36
37         // 回溯切割路径
38         List<Integer> res = new ArrayList<>();
39         backtrack(X, path, res);
40         Collections.sort(res);
41
42         // 输出
43         for (int i = 0; i < res.size(); i++) {
44             System.out.print(res.get(i));
```

```

45         if (i != res.size() - 1)
46             System.out.print(" ");
47     }
48     System.out.println();
49 }
50
51 // 回溯函数
52 private static void backtrack(int n, int[] path, List<Integer> res) {
53     if (path[n] == 0) {
54         res.add(n);
55     } else {
56         backtrack(path[n], path, res);
57         backtrack(n - path[n], path, res);
58     }
59 }
60 }

```

## Python

```
1  def solve(x):
2      dp = [(0, 0)] * (x + 1)
3      path = [0] * (x + 1)
4
5      for i in range(1, x + 1):
6          maxVal = i
7          minCut = 0
8          cutPos = 0
9
10         for j in range(1, i):
11             leftVal, leftCut = dp[j]
12             rightVal, rightCut = dp[i - j]
13             prod = leftVal * rightVal
14             cuts = leftCut + rightCut + 1
15
16
17             if prod > maxVal or (prod == maxVal and cuts < minCut):
18                 maxVal = prod
19                 minCut = cuts
20                 cutPos = j
21
22         dp[i] = (maxVal, minCut)
23         path[i] = cutPos
24
25
26     res = []
27     def backtrack(n):
28         if path[n] == 0:
29             res.append(n)
30         else:
31             backtrack(path[n])
32             backtrack(n - path[n])
33
34     backtrack(x)
35     print(" ".join(map(str, sorted(res))))
36
37
38 x = int(input())
39 solve(x)
```

## JavaScript



```
1  const readline = require('readline');
2
3  const rl = readline.createInterface({
4    input: process.stdin,
5    output: process.stdout
6  });
7
8  let input = [];
9  rl.on('line', line => {
10    input.push(line.trim());
11    if (input.length === 1) {
12      solve(parseInt(input[0]));
13      rl.close();
14    }
15  });
16
17  function solve(X) {
18    const dp = Array.from({ length: X + 1 }, () => [0, 0]);
19    const path = Array(X + 1).fill(0);
20
21    for (let i = 1; i <= X; i++) {
22      let maxVal = i;
23      let minCuts = 0;
24      let cutPos = 0;
25
26      for (let j = 1; j < i; j++) {
27        const [leftVal, leftCuts] = dp[j];
28        const [rightVal, rightCuts] = dp[i - j];
29        const prod = leftVal * rightVal;
30        const cuts = leftCuts + rightCuts + 1;
31
32        if (prod > maxVal || (prod === maxVal && cuts < minCuts)) {
33          maxVal = prod;
34          minCuts = cuts;
35          cutPos = j;
36        }
37      }
38
39      dp[i] = [maxVal, minCuts];
40      path[i] = cutPos;
41    }
42
43    const res = [];
44    function backtrack(n) {
```

```
46         if (path[n] === 0) {
47             res.push(n);
48         } else {
49             backtrack(path[n]);
50             backtrack(n - path[n]);
51         }
52     }
53
54     backtrack(X);
55     res.sort((a, b) => a - b);
56     console.log(res.join(' '));
57 }
```

**Go**

```
1  package main
2
3  import (
4      "bufio"
5      "fmt"
6      "os"
7      "sort"
8      "strconv"
9  )
10
11 func backtrack(n int, path []int, res *[]int) {
12     if path[n] == 0 {
13         *res = append(*res, n)
14     } else {
15         backtrack(path[n], path, res)
16         backtrack(n - path[n], path, res)
17     }
18 }
19
20 func main() {
21     reader := bufio.NewReader(os.Stdin)
22     line, _ := reader.ReadString('\n')
23     X, _ := strconv.Atoi(line[:len(line)-1])
24
25     type State struct {
26         value int
27         cuts  int
28     }
29
30     dp := make([]State, X+1)
31     path := make([]int, X+1)
32
33     for i := 1; i <= X; i++ {
34         maxVal := i
35         minCuts := 0
36         cutPos := 0
37
38         for j := 1; j < i; j++ {
39             left := dp[j]
40             right := dp[i-j]
41             prod := left.value * right.value
42             cuts := left.cuts + right.cuts + 1
43
44             if prod > maxVal || (prod == maxVal && cuts < minCuts) {
45                 maxVal = prod
```

```

46         minCuts = cuts
47         cutPos = j
48     }
49 }
50
51     dp[i] = State{maxVal, minCuts}
52     path[i] = cutPos
53 }
54
55     var res []int
56     backtrack(X, path, &res)
57     sort.Ints(res)
58
59     for i, v := range res {
60         fmt.Print(v)
61         if i != len(res)-1 {
62             fmt.Print(" ")
63         }
64     }
65     fmt.Println()
66 }

```

来自: [华为OD机试 2025 B卷 - 伐木工 \(C++ & Python & JAVA & JS & GO\)-CSDN博客](#)

# 华为OD机试 2025 B卷 - 查找接口成功率最优时间段 (C++ & Python & JAVA)

## 查找接口成功率最优时间段

华为OD机试真题目录点击查看: [华为OD机试2025B卷真题题库目录](#) | [机考题库](#) + [算法考点详解](#)

华为OD机试2025B卷 100分题型

### 题目描述

服务之间交换的接口成功率作为[服务调用](#)关键质量特性，某个时间段内的接口失败率使用一个数组表示，数组中每个元素都是单位时间内失败率数值，数组中的数值为0~100的整数，给定一个数值(minAverageLost)表示某个时间段内平均失败率容忍值，即平均失败率小于等于minAverageLost，找出数组中最长时间段，如果未找到则直接返回NULL。

### 输入描述

输入有两行内容，第一行{minAverageLost}，第二行为{数组}，数组元素通过空格(“ ”)分隔，minAverageLost及数组中元素取值范围为0~100的整数，数组元素的个数不会超过100个。

### 输出描述

找出[平均值](#)小于等于minAverageLost的最长时间段，输出数组下标对，格式{beginIndex}–{endIndx}(下标从0开始)，如果同时存在多个最长时间段，则输出多个下标对且下标对之间使用空格(“ ”)拼接，多个下标对按下标从小到大排序。

### 示例1

#### 输入

#### 输出

#### 说明

输入解释：minAverageLost=1，数组[0, 1, 2, 3, 4]前3个元素的平均值为1，因此数组第一个至第三个数组下标，即0-2

## 示例2

### 输入

### 输出

### 说明

输入解释：minAverageLost=2，数组[0, 0, 100, 2, 2, 99, 0, 2]通过计算小于等于2的最长时间段为：数组下标为0-1即[0, 0]，数组下标为3-4即[2, 2]，数组下标为6-7即[0, 2]，这三个部分都满足平均值小于等于2的要求，因此输出0-1 3-4 6-7

### 题解

思路：前缀和 基础算法运用

1. 预计算前缀和数组，使用 前缀和 数组减少求一个区间和的重复运算。
2. 双for循环从前往后确定两个区间的 起点和终点，判断这个区间的是否满足minAverageLost，并且是否为当前遍历过的最长区间？是的话加入结果候选集。
3. 输出最终最长区间的结果的满足条件的区间起点和终点。如果不存在满足条件的区间，输出NULL。

C++

```
1  #include<iostream>
2  #include<vector>
3  #include<string>
4  #include <utility>
5  #include <sstream>
6  #include<algorithm>
7  #include<map>
8  using namespace std;
9
10 int main() {
11     int minAverageLost;
12
13     cin >> minAverageLost;
14     map<int, vector<pair<int, int>>> mp;
15     vector<int> ans(100, 0);
16     int len = 0;
17     int tmp;
18
19     while (cin >> tmp) {
20         ans[len++] = tmp;
21     }
22     int currentLen = 0;
23     vector<int> prefix(len+ 1, 0);
24
25     for (int i = 1; i <= len; i++) {
26         prefix[i] = prefix[i-1] + ans[i-1];
27     }
28
29     for (int i = 0; i <= len; i++) {
30         for (int j = i + 1 ; j <= len; j++) {
31             if (j - i < currentLen) {
32                 continue;
33             }
34             int sum = prefix[j] - prefix[i];
35             if (sum * 1.0 / (j-i) <= minAverageLost){
36                 currentLen = max(currentLen, j - i);
37                 mp[currentLen].push_back({i, j-1});
38             }
39         }
40     }
41
42     if (currentLen == 0) {
43         cout<< "NULL";
44         return 0;
45     }
```

```
46
47     vector<pair<int, int>> maxLenV = mp[currentLen];
48     for (int i = 0; i < maxLenV.size(); i++) {
49         if (i != 0) {
50             cout << " ";
51         }
52         pair<int, int> p = maxLenV[i];
53         cout<< p.first << "-"<<p.second;
54     }
55     return 0;
56 }
```

## Java



```
1  import java.util.*;
2
3  public class Main {
4      public static void main(String[] args) {
5          Scanner scanner = new Scanner(System.in);
6
7          int minAverageLost = scanner.nextInt();
8          List<Integer> ans = new ArrayList<>();
9
10         while (scanner.hasNextInt()) {
11             ans.add(scanner.nextInt());
12         }
13         int len = ans.size();
14
15         if (len == 0) {
16             System.out.println("NULL");
17             return;
18         }
19
20
21         int[] prefix = new int[len + 1];
22         for (int i = 1; i <= len; i++) {
23             prefix[i] = prefix[i - 1] + ans.get(i - 1);
24         }
25
26         int currentLen = 0;
27         Map<Integer, List<int[]>> mp = new HashMap<>();
28
29
30         for (int i = 0; i <= len; i++) {
31             for (int j = i + 1; j <= len; j++) {
32                 if (j - i < currentLen) {
33                     continue;
34                 }
35                 int sum = prefix[j] - prefix[i];
36                 if ((double) sum / (j - i) <= minAverageLost) {
37                     currentLen = Math.max(currentLen, j - i);
38                     mp.putIfAbsent(currentLen, new ArrayList<>());
39                     mp.get(currentLen).add(new int[]{i, j - 1});
40                 }
41             }
42         }
43
44         if (currentLen == 0) {
45             System.out.println("NULL");
```

```

46         return;
47     }
48
49     List<int[]> maxLenV = mp.get(currentLen);
50     for (int i = 0; i < maxLenV.size(); i++) {
51         if (i != 0) System.out.print(" ");
52         System.out.print(maxLenV.get(i)[0] + "-" + maxLenV.get(i)[1]);
53     }
54 }
55 }

```

## Python

```
1  import sys
2
3  def main():
4      data = list(map(int, sys.stdin.read().split()))
5
6      min_average_lost = data[0]
7      ans = data[1:]
8      len_ans = len(ans)
9
10     if len_ans == 0:
11         print("NULL")
12         return
13
14
15     prefix = [0] * (len_ans + 1)
16     for i in range(1, len_ans + 1):
17         prefix[i] = prefix[i - 1] + ans[i - 1]
18
19     current_len = 0
20     mp = {}
21
22
23     for i in range(len_ans + 1):
24         for j in range(i + 1, len_ans + 1):
25             if j - i < current_len:
26                 continue
27             total = prefix[j] - prefix[i]
28             if total / (j - i) <= min_average_lost:
29                 current_len = max(current_len, j - i)
30                 mp.setdefault(current_len, []).append((i, j - 1))
31
32     if current_len == 0:
33         print("NULL")
34         return
35
36     result = ["{}-{}".format(start, end) for start, end in mp[current_len]]
37     print(" ".join(result))
38
39 if __name__ == "__main__":
40     main()
```

## JavaScript

```
1  const readline = require("readline");
2
3  const rl = readline.createInterface({
4    input: process.stdin,
5    output: process.stdout
6  });
7
8  let inputData = [];
9
10 rl.on("line", (line) => {
11   inputData.push(line.trim());
12   if (inputData.length === 2) rl.close();
13 });
14
15 rl.on("close", () => {
16   if (inputData.length < 2) {
17     console.log("NULL");
18     return;
19   }
20
21   const minAverageLost = parseInt(inputData[0]);
22   const ans = inputData[1].split(/\s+/).map(Number);
23   const len = ans.length;
24
25   if (len === 0) {
26     console.log("NULL");
27     return;
28   }
29
30
31   const prefix = new Array(len + 1).fill(0);
32   for (let i = 1; i <= len; i++) {
33     prefix[i] = prefix[i - 1] + ans[i - 1];
34   }
35
36   let currentLen = 0;
37   const mp = new Map();
38
39
40   for (let i = 0; i <= len; i++) {
41     for (let j = i + 1; j <= len; j++) {
42       if (j - i < currentLen) continue;
43       const sum = prefix[j] - prefix[i];
44       if (sum / (j - i) <= minAverageLost) {
45         currentLen = Math.max(currentLen, j - i);
```

```

46             if (!mp.has(currentLen)) mp.set(currentLen, []);
47             mp.get(currentLen).push([i, j - 1]);
48         }
49     }
50 }
51
52 if (currentLen === 0) {
53     console.log("NULL");
54     return;
55 }
56
57 const maxLenV = mp.get(currentLen);
58 console.log(maxLenV.map(([start, end]) => `${start}-${end}`).join("
59 "));
60 });

```

**Go**

```
1 package main
2
3 import (
4     "bufio"
5     "fmt"
6     "os"
7     "strconv"
8     "strings"
9 )
10
11 func main() {
12     scanner := bufio.NewScanner(os.Stdin)
13
14     scanner.Scan()
15     minAverageLost, _ := strconv.Atoi(scanner.Text())
16
17
18     if !scanner.Scan() {
19         fmt.Println("NULL")
20         return
21     }
22
23
24     numsStr := strings.Fields(scanner.Text())
25     var ans []int
26     for _, num := range numsStr {
27         val, _ := strconv.Atoi(num)
28         ans = append(ans, val)
29     }
30
31     lenAns := len(ans)
32     if lenAns == 0 {
33         fmt.Println("NULL")
34         return
35     }
36
37
38     prefix := make([]int, lenAns+1)
39     for i := 1; i <= lenAns; i++ {
40         prefix[i] = prefix[i-1] + ans[i-1]
41     }
42
43     currentLen := 0
44     mp := make(map[int][][]int)
```

```

46
47
48     for i := 0; i <= lenAns; i++ {
49         for j := i + 1; j <= lenAns; j++ {
50             if j-i < currentLen {
51                 continue
52             }
53             sum := prefix[j] - prefix[i]
54             if float64(sum)/float64(j-i) <= float64(minAverageLost) {
55                 currentLen = max(currentLen, j-i)
56                 mp[currentLen] = append(mp[currentLen], []int{i, j - 1})
57             }
58         }
59     }
60
61     if currentLen == 0 {
62         fmt.Println("NULL")
63         return
64     }
65
66
67     var result []string
68     for _, pair := range mp[currentLen] {
69         result = append(result, fmt.Sprintf("%d-%d", pair[0], pair[1]))
70     }
71     fmt.Println(strings.Join(result, " "))
72 }
73
74
75 func max(a, b int) int {
76     if a > b {
77         return a
78     }
79     return b
80 }

```

来自: [华为OD机试 2025 B卷 - 查找接口成功率最优时间段 \(C++ & Python & JAVA & JS & GO\)-CSDN博客](#)

# 华为OD机试 2025 B卷 - 抢7游戏 (C++ & Python & JAVA & JS &

## 抢7游戏

华为OD机试真题目录点击查看: [华为OD机试2025B卷真题题库目录](#) | [机考题库](#) + [算法考点详解](#)

华为OD机试2025B卷 100分题型

## 题目描述

A、B两个人玩抢7游戏，[游戏规则](#)为：

A先报一个起始数字  $X$  ( $10 \leq \text{起始数字} \leq 10000$ )，B报下一个数字  $Y$  ( $X - Y < 3$ )，A再报一个数字  $Z$  ( $Y - Z < 3$ )，以此类推，直到其中一个抢到7，抢到7即为胜者；

在B赢得比赛的情况下，一共有多少种组合？

## 输入描述

起始数字  $M$ 。  $1 \leq M \leq 10000$

## 输出描述

B能赢得比赛的组合次数

## 用例1

### 输入

### 输出

## 题解

思路：动态规划

- 其实这道题的思路一道经典题 [爬楼梯方案数](#) 非常相似。定义  $dpA[]$  其中  $dpA[i]$  表示A含数字i的方案数，定义  $dpB[]$  其中  $dpB[j]$  表示B喊B的方案数。
- 由题意可以  $dpA[i]$  的值由  $dpB[i+1]$ 和 $dpB[i+2]$ 的决定。  $dpB[j]$  类似。所以可以直接得出状态转移方程
  - $dpB[i] = dpA[i + 1] + dpA[i + 2]$



- $dpA[i] = dpB[i + 1] + dpB[i + 2]$

3. 明白2的状态方程之后，从m枚举到7就能直到，B能赢得比赛的组合数。对应结果为  $dpB[7]$  .

额外注意，m的值[1,10000]，进行状态转移过程会出现非常大的数，经典的 大数相加 问题。对于没有内置大数的编程语言，需要手动定义函数实现(字符串相加)。

**C++**

```
1  #include<iostream>
2  #include<vector>
3  #include<string>
4  #include <utility>
5  #include <sstream>
6  #include<algorithm>
7  using namespace std;
8
9
10 // 实现两个大整数的加法
11 string addBigNumbers(const string& num1, const string& num2) {
12     string res;
13     // 进位
14     int carry = 0;
15
16     int i = num1.size() - 1;
17     int j = num2.size() - 1;
18
19     // 从低位到高位逐位相加
20     while (i >= 0 || j >= 0 || carry) {
21         int digit1 = i >= 0 ? (num1[i--] - '0') : 0;
22         int digit2 = j >= 0 ? (num2[j--] - '0') : 0;
23         int sum = digit1 + digit2 + carry;
24         res += (sum % 10 + '0');
25         carry = sum / 10;
26     }
27
28     reverse(res.begin(), res.end());
29     return res;
30 }
31
32 int main() {
33     int m;
34     cin >> m;
35     // i 由 i+1 i+2 累加得来，所以定义长度为m + 2
36     vector<string> dpA(m + 2, "0");
37     dpA[m] = "1";
38     vector<string> dpB(m + 2, "0");
39     for (int i = m - 1; i >= 7; i--) {
40         // B得出的方案数 由 A叫 i + 1 + 叫 i+2 的方案数得来
41         dpB[i] = addBigNumbers(dpA[i + 1], dpA[i + 2]);
42         // A得出的方案数 由 B叫 i + 1 + 叫 i+2 的方案数得来
43         dpA[i] = addBigNumbers(dpB[i + 1], dpB[i + 2]);
44     }
45     cout << dpB[7];
```

```
46     return 0;
47 }
```

## JAVA

▼ Plain Text

```
1  import java.math.BigInteger;
2  import java.util.Scanner;
3
4  public class Main {
5      public static void main(String[] args) {
6          Scanner scanner = new Scanner(System.in);
7          int m = scanner.nextInt();
8
9          // i 由 i+1 i+2 累加得来, 所以定义长度为 m + 2
10         BigInteger[] dpA = new BigInteger[m + 2];
11         BigInteger[] dpB = new BigInteger[m + 2];
12
13         for (int i = 0; i < m + 2; i++) {
14             dpA[i] = BigInteger.ZERO;
15             dpB[i] = BigInteger.ZERO;
16         }
17
18         dpA[m] = BigInteger.ONE;
19
20         for (int i = m - 1; i >= 7; i--) {
21             // B得出的方案数 由 A叫 i + 1 + 叫 i+2 的方案数得来
22             dpB[i] = dpA[i + 1].add(dpA[i + 2]);
23             // A得出的方案数 由 B叫 i + 1 + 叫 i+2 的方案数得来
24             dpA[i] = dpB[i + 1].add(dpB[i + 2]);
25         }
26
27         System.out.println(dpB[7]);
28     }
29 }
```

## Python

```
1  def main():
2      m = int(input())
3
4
5      dpA = [0] * (m + 2)
6      dpB = [0] * (m + 2)
7
8      dpA[m] = 1
9
10
11     for i in range(m - 1, 6, -1):
12
13         dpB[i] = dpA[i + 1] + dpA[i + 2]
14
15         dpA[i] = dpB[i + 1] + dpB[i + 2]
16
17     print(dpB[7])
18
19 if __name__ == "__main__":
20     main()
```

## JavaScript

```
1  const readline = require('readline');
2
3  const rl = readline.createInterface({
4    input: process.stdin,
5    output: process.stdout
6  });
7
8  let inputLines = [];
9
10 rl.on('line', function (line) {
11   inputLines.push(line);
12 }).on('close', function () {
13   let m = parseInt(inputLines[0]);
14
15
16   const dpA = Array(m + 2).fill(0n);
17   const dpB = Array(m + 2).fill(0n);
18
19   dpA[m] = 1n;
20
21   for (let i = m - 1; i >= 7; i--) {
22
23     dpB[i] = dpA[i + 1] + dpA[i + 2];
24
25     dpA[i] = dpB[i + 1] + dpB[i + 2];
26   }
27
28   console.log(dpB[7].toString());
29 });
```

**Go**

```
1 package main
2
3 import (
4     "fmt"
5     "math/big"
6 )
7
8 func main() {
9     var m int
10    fmt.Scan(&m)
11
12
13    dpA := make([]*big.Int, m+2)
14    dpB := make([]*big.Int, m+2)
15
16    for i := 0; i < m+2; i++ {
17        dpA[i] = big.NewInt(0)
18        dpB[i] = big.NewInt(0)
19    }
20
21    dpA[m] = big.NewInt(1)
22
23    for i := m - 1; i >= 7; i-- {
24
25        dpB[i] = new(big.Int).Add(dpA[i+1], dpA[i+2])
26
27        dpA[i] = new(big.Int).Add(dpB[i+1], dpB[i+2])
28    }
29
30    fmt.Println(dpB[7].String())
31 }
```

来自: [华为OD机试 2025 B卷 – 抢7游戏 \(C++ & Python & JAVA & JS & GO\)-CSDN博客](#)

## 抢7游戏

[华为OD机试](#)真题目录点击查看: [华为OD机试2025B卷真题题库目录](#) | [机考题库](#) + [算法考点详解](#)

华为OD机试2025B卷 100分题型

## 题目描述

A、B两个人玩抢7游戏，[游戏规则](#)为：

A先报一个起始数字  $X$  ( $10 \leq \text{起始数字} \leq 10000$ )，B报下一个数字  $Y$  ( $X - Y < 3$ )，A再报一个数字  $Z$  ( $Y - Z < 3$ )，以此类推，直到其中一个抢到7，抢到7即为胜者；

在B赢得比赛的情况下，一共有多少种组合？

## 输入描述

起始数字  $M$ 。  $1 \leq M \leq 10000$

## 输出描述

B能赢得比赛的组合次数

## 用例1

### 输入

### 输出

## 题解

思路： [动态规划](#)

1. 其实这道题的思路一道经典题 [爬楼梯方案数](#) 非常相似。定义  $dpA[]$  其中  $dpA[i]$  表示A含数字i的方案数，定义  $dpB[]$  其中  $dpB[j]$  表示B喊B的方案数。
2. 由题意可以  $dpA[i]$  的值由  $dpB[i+1]$  和  $dpB[i+2]$  的决定。  $dpB[j]$  类似。所以可以直接得出状态转移方程
  - $dpB[i] = dpA[i + 1] + dpA[i + 2]$
  - $dpA[i] = dpB[i + 1] + dpB[i + 2]$
3. 明白2的状态方程之后，从m枚举到7就能直到，B能赢得比赛的组合数。对应结果为  $dpB[7]$ 。

额外注意，m的值[1,10000]，进行状态转移过程会出现非常大的数，经典的 [大数相加](#) 问题。对于没有内置大数的编程语言，需要手动定义函数实现(字符串相加)。

### C++

```
1  #include<iostream>
2  #include<vector>
3  #include<string>
4  #include <utility>
5  #include <sstream>
6  #include<algorithm>
7  using namespace std;
8
9
10 // 实现两个大整数的加法
11 string addBigNumbers(const string& num1, const string& num2) {
12     string res;
13     // 进位
14     int carry = 0;
15
16     int i = num1.size() - 1;
17     int j = num2.size() - 1;
18
19     // 从低位到高位逐位相加
20     while (i >= 0 || j >= 0 || carry) {
21         int digit1 = i >= 0 ? (num1[i--] - '0') : 0;
22         int digit2 = j >= 0 ? (num2[j--] - '0') : 0;
23         int sum = digit1 + digit2 + carry;
24         res += (sum % 10 + '0');
25         carry = sum / 10;
26     }
27
28     reverse(res.begin(), res.end());
29     return res;
30 }
31
32 int main() {
33     int m;
34     cin >> m;
35     // i 由 i+1 i+2 累加得来，所以定义长度为m + 2
36     vector<string> dpA(m + 2, "0");
37     dpA[m] = "1";
38     vector<string> dpB(m + 2, "0");
39     for (int i = m - 1; i >= 7; i--) {
40         // B得出的方案数 由 A叫 i + 1 + 叫 i+2 的方案数得来
41         dpB[i] = addBigNumbers(dpA[i + 1], dpA[i + 2]);
42         // A得出的方案数 由 B叫 i + 1 + 叫 i+2 的方案数得来
43         dpA[i] = addBigNumbers(dpB[i + 1], dpB[i + 2]);
44     }
45     cout << dpB[7];
```



```
46     return 0;
47 }
```

## JAVA

```
1  import java.math.BigInteger;
2  import java.util.Scanner;
3
4  public class Main {
5      public static void main(String[] args) {
6          Scanner scanner = new Scanner(System.in);
7          int m = scanner.nextInt();
8
9          // i 由 i+1 i+2 累加得来, 所以定义长度为 m + 2
10         BigInteger[] dpA = new BigInteger[m + 2];
11         BigInteger[] dpB = new BigInteger[m + 2];
12
13         for (int i = 0; i < m + 2; i++) {
14             dpA[i] = BigInteger.ZERO;
15             dpB[i] = BigInteger.ZERO;
16         }
17
18         dpA[m] = BigInteger.ONE;
19
20         for (int i = m - 1; i >= 7; i--) {
21             // B得出的方案数 由 A叫 i + 1 + 叫 i+2 的方案数得来
22             dpB[i] = dpA[i + 1].add(dpA[i + 2]);
23             // A得出的方案数 由 B叫 i + 1 + 叫 i+2 的方案数得来
24             dpA[i] = dpB[i + 1].add(dpB[i + 2]);
25         }
26
27         System.out.println(dpB[7]);
28     }
29 }
```

## Python

```
1  def main():
2      m = int(input())
3
4
5      dpA = [0] * (m + 2)
6      dpB = [0] * (m + 2)
7
8      dpA[m] = 1
9
10
11     for i in range(m - 1, 6, -1):
12
13         dpB[i] = dpA[i + 1] + dpA[i + 2]
14
15         dpA[i] = dpB[i + 1] + dpB[i + 2]
16
17     print(dpB[7])
18
19 if __name__ == "__main__":
20     main()
```

## JavaScript

```
1  const readline = require('readline');
2
3  const rl = readline.createInterface({
4    input: process.stdin,
5    output: process.stdout
6  });
7
8  let inputLines = [];
9
10 rl.on('line', function (line) {
11   inputLines.push(line);
12 }).on('close', function () {
13   let m = parseInt(inputLines[0]);
14
15
16   const dpA = Array(m + 2).fill(0n);
17   const dpB = Array(m + 2).fill(0n);
18
19   dpA[m] = 1n;
20
21   for (let i = m - 1; i >= 7; i--) {
22
23     dpB[i] = dpA[i + 1] + dpA[i + 2];
24
25     dpA[i] = dpB[i + 1] + dpB[i + 2];
26   }
27
28   console.log(dpB[7].toString());
29 });
```

Go

```
1  package main
2
3  import (
4      "fmt"
5      "math/big"
6  )
7
8  func main() {
9      var m int
10     fmt.Scan(&m)
11
12
13     dpA := make([]*big.Int, m+2)
14     dpB := make([]*big.Int, m+2)
15
16     for i := 0; i < m+2; i++ {
17         dpA[i] = big.NewInt(0)
18         dpB[i] = big.NewInt(0)
19     }
20
21     dpA[m] = big.NewInt(1)
22
23     for i := m - 1; i >= 7; i-- {
24
25         dpB[i] = new(big.Int).Add(dpA[i+1], dpA[i+2])
26
27         dpA[i] = new(big.Int).Add(dpB[i+1], dpB[i+2])
28     }
29
30     fmt.Println(dpB[7].String())
31 }
```

来自: [华为OD机试 2025 B卷 - 抢7游戏 \(C++ & Python & JAVA & JS & GO\)-CSDN博客](#)

# 华为OD 机试 2025 B卷 - 数组二叉树 (C++ & Python & JAVA & JS)

## 数组二叉树

华为OD机试真题目录点击查看: [华为OD机试2025B卷真题题库目录](#) | [机考题库](#) + [算法考点详解](#)

华为OD机试2025B卷 100分题型

## 题目描述

二叉树也可以用数组来存储, 给定一个数组, 树的根节点的值存储在下标1, 对于存储在下标N的节点, 它的左子节点和右子节点分别存储在下标 $2N$ 和 $2N+1$ , 并且我们用值-1代表一个节点为空。

给定一个数组存储的二叉树, 试求从根节点到最小的叶子节点的路径, 路径由节点的值组成。

## 输入描述

输入一行为数组的内容, 数组的每个元素都是正整数, 元素间用空格分隔。

注意第一个元素即为根节点的值, 即数组的第N个元素对应下标N, 下标0在树的表示中没有使用, 所以我们省略了。

输入的树最多为7层。

## 输出描述

输出从根节点到最小叶子节点的路径上, 各个节点的值, 由空格分隔, 用例保证最小叶子节点只有一个。

## 用例1

### 输入

### 输出

### 说明

最小叶子节点的路径为3 7 2。

## 示例二

### 输入



Plain Text

```
1 5 9 8 -1 -1 7 -1 -1 -1 -1 -1 6
```

## 输出

## 说明



Plain Text

- 1 最小叶子节点的路径为5 8 7 6，注意数组仅存储至最后一个非空节点，故不包含节点“7”右子节点的-1。

## 题解

思路： 逻辑分析题 + 模拟

1. 题目要求 从根节点到最小叶子节点的路径，这题如果从根节点出发去寻找路径需要考虑的情况蛮多了，但是如果你先找到最小叶子节点，再往上递推到根节点就会非常容易。 转变思路
2. 接收输入。从后往前遍历找到最小叶子节点位置 `pos`。
3. 循环迭代，获取从最小叶子节点到根节点的路径。在数组中对于 `position` 的父节点位置为 `(position - 1) / 2`。
4. 经过3处理之后，输出得到的路径即可。

C++

```
1  #include <cstdint>
2  #include<iostream>
3  #include<vector>
4  #include<string>
5  #include <utility>
6  #include <sstream>
7  #include<algorithm>
8  #include <cmath>
9  #include <climits>
10 using namespace std;
11
12 // 计算完全二叉树的层数
13 int compute_height(int N) {
14     return ceil(log2(N + 1));
15 }
16
17 bool judege(vector<int> ans, int pos) {
18     int n = ans.size();
19     return (pos >= n || ans[pos] == -1);
20 }
21
22 int main() {
23     vector<int> ans;
24     int tmp;
25     while (cin >> tmp) {
26         ans.push_back(tmp);
27     }
28
29     vector<int> res;
30     int n = ans.size();
31     // 最小叶子节点值位置
32     int pos = -1;
33     // 最小叶子节点值位置
34     int minValue = INT_MAX;
35
36     for (int i = 0; i < n; i++) {
37         int leftIndex = 2 * (i+1) -1;
38         int rightIndex = 2 * (i+1);
39         if (ans[i] == -1) {
40             continue;
41         }
42         // 判断是否是叶子节点
43         if (judege(ans, leftIndex) && judege(ans, rightIndex)) {
44             if (ans[i] < minValue) {
45                 pos = i;
```

```

46             minValue = ans[i];
47         }
48     }
49 }
50 // 添加叶子节点到根节点的值
51 while (pos != 0) {
52     res.push_back(ans[pos]);
53     pos = (pos-1) / 2;
54 }
55 res.push_back(ans[0]);
56
57 for (int i = res.size()-1; i >= 0; i--) {
58     cout << res[i];
59     if (i != 0) {
60         cout << " ";
61     }
62 }
63 return 0;
64 }

```

## JAVA



```
1  import java.util.*;
2
3  public class Main {
4      // 计算完全二叉树的层数
5      public static int computeHeight(int N) {
6          return (int) Math.ceil(Math.log(N + 1) / Math.log(2));
7      }
8
9      public static boolean judge(List<Integer> ans, int pos) {
10         return pos >= ans.size() || ans.get(pos) == -1;
11     }
12
13     public static void main(String[] args) {
14         Scanner sc = new Scanner(System.in);
15         List<Integer> ans = new ArrayList<>();
16
17         // 读取输入
18         while (sc.hasNextInt()) {
19             ans.add(sc.nextInt());
20         }
21
22         List<Integer> res = new ArrayList<>();
23         int n = ans.size();
24
25         int pos = -1;
26         int minValue = Integer.MAX_VALUE;
27
28         // 查找最小的叶子节点值位置
29         for (int i = 0; i < n; i++) {
30             int leftIndex = 2 * (i + 1) - 1;
31             int rightIndex = 2 * (i + 1);
32             if (ans.get(i) == -1) {
33                 continue;
34             }
35
36             // 判断是否是叶子节点
37             if (judge(ans, leftIndex) && judge(ans, rightIndex)) {
38                 if (ans.get(i) < minValue) {
39                     pos = i;
40                     minValue = ans.get(i);
41                 }
42             }
43         }
44
45         // 从叶子节点到根节点
```

```
46         while (pos != 0) {
47             res.add(ans.get(pos));
48             pos = (pos - 1) / 2;
49         }
50         res.add(ans.get(0));
51
52         // 输出结果
53         for (int i = res.size() - 1; i >= 0; i--) {
54             System.out.print(res.get(i));
55             if (i != 0) {
56                 System.out.print(" ");
57             }
58         }
59
60         sc.close();
61     }
62 }
```

## Python

```
1  import math
2
3
4  def compute_height(N):
5      return math.ceil(math.log2(N + 1))
6
7
8  def judge(ans, pos):
9      return pos >= len(ans) or ans[pos] == -1
10
11 def main():
12     ans = []
13
14
15     try:
16
17         inputs = input().split()
18         for num in inputs:
19             ans.append(int(num))
20     except EOFError:
21         pass
22
23     res = []
24     n = len(ans)
25
26
27     pos = -1
28     min_value = float('inf')
29
30
31     for i in range(n):
32         left_index = 2 * (i + 1) - 1
33         right_index = 2 * (i + 1)
34         if ans[i] == -1:
35             continue
36
37
38         if judge(ans, left_index) and judge(ans, right_index):
39             if ans[i] < min_value:
40                 pos = i
41                 min_value = ans[i]
42
43
44     while pos != 0:
45         res.append(ans[pos])
```

```
46         pos = (pos - 1) // 2
47         res.append(ans[0])
48
49
50         print(" ".join(map(str, res[::-1])))
51
52     if __name__ == "__main__":
53         main()
```

## JavaScript

```
1  function computeHeight(N) {
2      return Math.ceil(Math.log2(N + 1));
3  }
4
5
6  function judge(ans, pos) {
7      return pos >= ans.length || ans[pos] === -1;
8  }
9
10 function main() {
11
12     let input = '';
13     let stdin = process.stdin;
14     stdin.setEncoding('utf-8');
15     stdin.on('data', function (data) {
16         input += data;
17     });
18
19     stdin.on('end', function () {
20
21         const ans = input.trim().split(/\s+/).map(Number);
22
23         let res = [];
24         const n = ans.length;
25
26
27         let pos = -1;
28         let minValue = Infinity;
29
30
31         for (let i = 0; i < n; i++) {
32             let leftIndex = 2 * (i + 1) - 1;
33             let rightIndex = 2 * (i + 1);
34
35             if (ans[i] === -1) {
36                 continue;
37             }
38
39
40             if (judge(ans, leftIndex) && judge(ans, rightIndex)) {
41                 if (ans[i] < minValue) {
42                     pos = i;
43                     minValue = ans[i];
44                 }
45             }
```

```
46         }
47
48
49         while (pos !== 0) {
50             res.push(ans[pos]);
51             pos = Math.floor((pos - 1) / 2);
52         }
53         res.push(ans[0]);
54
55
56         console.log(res.reverse().join(' '));
57     });
58 }
59
60 main();
```

**Go**

```
1  package main
2
3  import (
4      "bufio"
5      "fmt"
6      "math"
7      "os"
8      "strconv"
9      "strings"
10 )
11
12
13
14
15 func judge(ans []int, pos int) bool {
16     return pos >= len(ans) || ans[pos] == -1
17 }
18
19 func main() {
20     scanner := bufio.NewScanner(os.Stdin)
21
22     scanner.Scan()
23     input := scanner.Text()
24     inputs := strings.Fields(input)
25
26
27     var ans []int
28     for _, val := range inputs {
29         num, err := strconv.Atoi(val)
30         if err != nil {
31             fmt.Println("输入无效")
32             return
33         }
34         ans = append(ans, num)
35     }
36
37     var res []int
38     pos := -1
39     minValue := math.MaxInt
40
41
42     for i := 0; i < len(ans); i++ {
43         leftIndex := 2*(i+1) - 1
44         rightIndex := 2*(i+1)
45         if ans[i] == -1 {
```

```

46         continue
47     }
48
49     if judge(ans, leftIndex) && judge(ans, rightIndex) {
50         if ans[i] < minValue {
51             pos = i
52             minValue = ans[i]
53         }
54     }
55 }
56
57
58 for pos != 0 {
59     res = append(res, ans[pos])
60     pos = (pos - 1) / 2
61 }
62 res = append(res, ans[0])
63
64
65 for i := len(res) - 1; i >= 0; i-- {
66     fmt.Print(res[i])
67     if i != 0 {
68         fmt.Print(" ")
69     }
70 }
71 }

```

来自: [华为OD 机试 2025 B卷 – 数组二叉树 \(C++ & Python & JAVA & JS & GO\)-CSDN博客](#)



# 华为OD 机试 2025 B卷 - 路灯照明问题 (C++ & Python & JAVA & JS)

## 路灯照明问题

华为OD机试真题目录点击查看: [华为OD机试2025B卷真题题库目录](#) | [机考题库](#) + [算法考点详解](#)

华为OD机试2025B卷 100分题型

## 题目描述

在一条笔直的公路上安装了N个路灯，从位置0开始安装，路灯之间间距固定为100米。  
每个路灯都有自己的照明半径，请计算第一个路灯和最后一个路灯之间，无法照明的区间的长度和。

## 输入描述

第一行为一个数N，表示路灯个数， $1 \leq N \leq 100000$

第二行为N个空格分隔的数，表示路灯的照明半径， $1 \leq \text{照明半径} \leq 100000 \times 100$

## 输出描述

第一个路灯和最后一个路灯之间，无法照明的区间的长度和。

## 用例1

### 输入

### 输出

### 说明

路灯1覆盖0-50，路灯2覆盖50-100，路灯1和路灯2之间(0米-100米)无未覆盖的区间。

## 用例2

### 输入

## 输出

## 说明

| [170,180],[220,230], 两个未覆盖的区间, 总里程为20

## 题解

思路: 区间合并

1. 通过路灯数量, 可以得出第一个路灯和最后一个路灯的总距离。
2. 通过每一个路灯的照明半径  $r$  可以得出, 这个路灯的覆盖区间. 例如现在枚举的路灯为 $i$ , 照明长度为  $[i * 100 - r, i * 100 + r]$ 。
3. 通过第二步可以得出  $n$  的区间, 接下来进行区间合并就行. 注意递归合并区间。
4. 合并区间之后, 使用总距离 - 每个合并 覆盖区间长度 就是结果。

C++

```
1  #include<iostream>
2  #include<vector>
3  #include<string>
4  #include <utility>
5  #include <sstream>
6  #include<algorithm>
7  #include<stack>
8  using namespace std;
9
10 int main() {
11     int n ;
12     cin >> n;
13     vector<int> ans(n);
14     stack<pair<int,int>> stk;
15     for (int i = 0; i < n; i++) {
16         cin >> ans[i];
17         int left = (i * 100) - ans[i];
18         int right = (i*100) + ans[i];
19         // 递归区间合并
20         while (!stk.empty() && stk.top().second >= left) {
21             pair<int,int> tmp = stk.top();
22             stk.pop();
23             left = min(tmp.first, left);
24             right = max(tmp.second, right);
25         }
26         stk.push({left, right});
27     }
28     int res = 0;
29     // 计算多个区间中空白照明长度
30     while (stk.size() != 1) {
31         pair<int,int> top = stk.top();
32         stk.pop();
33         res += top.first - stk.top().second;
34     }
35
36     cout << res;
37 }
```

## JAVA

```
1  import java.util.*;
2
3  public class Main {
4      public static void main(String[] args) {
5          Scanner scanner = new Scanner(System.in);
6
7          // 读取整数 n
8          int n = scanner.nextInt();
9          int[] ans = new int[n];
10         Stack<int[]> stk = new Stack<>();
11
12         // 读取 n 个数并计算区间合并
13         for (int i = 0; i < n; i++) {
14             ans[i] = scanner.nextInt();
15             int left = (i * 100) - ans[i];
16             int right = (i * 100) + ans[i];
17
18             // 递归区间合并
19             while (!stk.isEmpty() && stk.peek()[1] >= left) {
20                 int[] tmp = stk.pop();
21                 left = Math.min(tmp[0], left);
22                 right = Math.max(tmp[1], right);
23             }
24             stk.push(new int[]{left, right});
25         }
26
27         // 计算多个区间中空白照明长度
28         int res = 0;
29         while (stk.size() > 1) {
30             int[] top = stk.pop();
31             res += top[0] - stk.peek()[1];
32         }
33
34         // 输出结果
35         System.out.println(res);
36     }
37 }
```

## Python

```
1  import sys
2
3  def main():
4
5      n = int(sys.stdin.readline().strip())
6      ans = list(map(int, sys.stdin.readline().strip().split()))
7      stk = []
8
9
10     for i in range(n):
11         left = (i * 100) - ans[i]
12         right = (i * 100) + ans[i]
13
14
15         while stk and stk[-1][1] >= left:
16             tmp = stk.pop()
17             left = min(tmp[0], left)
18             right = max(tmp[1], right)
19         stk.append((left, right))
20
21
22     res = 0
23     while len(stk) > 1:
24         top = stk.pop()
25         res += top[0] - stk[-1][1]
26
27
28     print(res)
29
30 if __name__ == "__main__":
31     main()
```

## JavaScript

```
1  const readline = require("readline");
2
3  const rl = readline.createInterface({
4    input: process.stdin,
5    output: process.stdout
6  });
7
8  let inputLines = [];
9
10 rl.on("line", (line) => {
11   inputLines.push(line);
12 }).on("close", () => {
13
14   let n = parseInt(inputLines[0]);
15   let ans = inputLines[1].split(" ").map(Number);
16   let stk = [];
17
18   for (let i = 0; i < n; i++) {
19     let left = (i * 100) - ans[i];
20     let right = (i * 100) + ans[i];
21
22     while (stk.length > 0 && stk[stk.length - 1][1] >= left) {
23       let tmp = stk.pop();
24       left = Math.min(tmp[0], left);
25       right = Math.max(tmp[1], right);
26     }
27     stk.push([left, right]);
28   }
29
30   let res = 0;
31   while (stk.length > 1) {
32     let top = stk.pop();
33     res += top[0] - stk[stk.length - 1][1];
34   }
35
36   console.log(res);
37 });
```

Go

```
1 package main
2
3 import (
4     "fmt"
5 )
6
7 func main() {
8     var n int
9     fmt.Scan(&n)
10
11     ans := make([]int, n)
12     for i := 0; i < n; i++ {
13         fmt.Scan(&ans[i])
14     }
15
16     type Interval struct {
17         left int
18         right int
19     }
20
21     stack := []Interval{}
22     for i := 0; i < n; i++ {
23         left := i*100 - ans[i]
24         right := i*100 + ans[i]
25
26         for len(stack) > 0 && stack[len(stack)-1].right >= left {
27             top := stack[len(stack)-1]
28             stack = stack[:len(stack)-1]
29             left = min(top.left, left)
30             right = max(top.right, right)
31         }
32         stack = append(stack, Interval{left, right})
33     }
34
35     res := 0
36
37     for i := 1; i < len(stack); i++ {
38         res += stack[i].left - stack[i-1].right
39     }
40
41     fmt.Println(res)
42 }
43
44
45 func min(a, b int) int {
```

```
46     if a < b {
47         return a
48     }
49     return b
50 }
51
52 func max(a, b int) int {
53     if a > b {
54         return a
55     }
56     return b
57 }
```

来自: [华为OD 机试 2025 B卷 – 路灯照明问题 \(C++ & Python & JAVA & JS & GO\)–CSDN博客](#)



# 华为OD机试 2025 B卷 - 最佳投资方式 (C++ & Python & JAVA & JS)

## 最佳投资方式 / 虚拟理财游戏

华为OD机试真题目录点击查看: [华为OD机试2025B卷真题题库目录](#) | [机考题库](#) + [算法考点详解](#)

华为OD机试2025B卷 100分题型

### 题目描述

在一款虚拟游戏中生活，你必须进行投资以增强在虚拟游戏中的资产以免被淘汰出局。

现有一家Bank，它提供有若干理财产品  $m$  个，风险及投资回报不同，你有  $N$ （元）进行投资，能接收的总风险值为 $X$ 。

你要在可接受范围内选择最优的投资方式获得最大回报。

备注：

- 在虚拟游戏中，每项投资风险值相加为总风险值；
- 在虚拟游戏中，最多只能投资2个理财产品；
- 在虚拟游戏中，最小单位为整数，不能拆分为小数；
- 投资额\*回报率=投资回报

### 输入描述

第一行：

- 产品数（取值范围[1,20]）
- 总投资额（整数，取值范围[1, 10000]）
- 可接受的总风险（整数，取值范围[1,200]）

第二行：产品[投资回报率](#)序列，输入为整数，取值范围[1,60]

第三行：产品风险值序列，输入为整数，取值范围[1, 100]

第四行：最大投资额度序列，输入为整数，取值范围[1, 10000]

### 输出描述

每个产品的投资额序列

### 示例1

输入

```
1  5 100 10
2  10 20 30 40 50
3  3 4 5 6 10
4  20 30 20 40 30
```

## 输出

## 说明

投资第二项30个单位，第四项40个单位，总的投资风险为两项相加为 $4+6=10$

## 题解

思路：模拟

1. 只需要考虑两种投资方式

a. 只投资一种产品。

b. 组合投资，两两组合产品进行投资，优先把金额投资到回报率高的产品上。

2. 结果为上述两种情况中出现的回报额最大的组合情况。尝试一种或者两种组合需要考虑以下因素。

◦ 投资产品风险和是否大于  $X$ ，超过则说明组合不合法。

◦ 两种组合情况下，优先将资金投入回报大的商品。

3. 通过循环匹配不同组合方案，记录其中能得到最大投资回报的方案就是结果，按题目要求格式输出即可。

**C++**

```
1  #include <iostream>
2  #include <vector>
3  #include <algorithm>
4  #include <sstream>
5  #include<iterator>
6
7  using namespace std;
8
9  vector<int> readIntArray() {
10     string line;
11     getline(cin, line);
12     istringstream iss(line);
13     return vector<int>(istream_iterator<int>(iss), {});
14 }
15
16 int main() {
17     vector<int> input = readIntArray();
18     int m = input[0], N = input[1], X = input[2];
19     vector<int> returns = readIntArray();
20     vector<int> risks = readIntArray();
21     vector<int> maxInvestments = readIntArray();
22
23     int maxReturn = 0;
24     vector<int> bestInvestments(m, 0);
25
26     for (int i = 0; i < m; i++) {
27         if (risks[i] > X) continue;
28
29         int investI = min(N, maxInvestments[i]);
30         int retI = investI * returns[i];
31         if (retI > maxReturn) {
32             maxReturn = retI;
33             fill(bestInvestments.begin(), bestInvestments.end(), 0);
34             bestInvestments[i] = investI;
35         }
36
37         for (int j = i + 1; j < m; j++) {
38             if (risks[i] + risks[j] > X) continue;
39
40             int investI, investJ;
41
42             if (returns[i] > returns[j]) {
43                 investI = min(N, maxInvestments[i]);
44                 investJ = min(N - investI, maxInvestments[j]);
45             } else {
```

```

46         investJ = min(N, maxInvestments[j]);
47         investI = min(N - investJ, maxInvestments[i]);
48     }
49
50     int retPair = investI * returns[i] + investJ * returns[j];
51     if (retPair > maxReturn) {
52         maxReturn = retPair;
53         fill(bestInvestments.begin(), bestInvestments.end(), 0);
54         bestInvestments[i] = investI;
55         bestInvestments[j] = investJ;
56     }
57 }
58 }
59
60 for (int investment : bestInvestments) {
61     cout << investment << " ";
62 }
63 cout << endl;
64
65 return 0;
66 }

```

## Java

```
1  import java.util.*;
2
3  public class Main {
4
5      private static List<Integer> readIntArray(Scanner scanner) {
6          String[] tokens = scanner.nextLine().split(" ");
7          List<Integer> numbers = new ArrayList<>();
8          for (String token : tokens) {
9              numbers.add(Integer.parseInt(token));
10         }
11         return numbers;
12     }
13
14     public static void main(String[] args) {
15         Scanner scanner = new Scanner(System.in);
16
17         List<Integer> input = readIntArray(scanner);
18         int m = input.get(0), N = input.get(1), X = input.get(2);
19
20
21         List<Integer> returns = readIntArray(scanner);
22         List<Integer> risks = readIntArray(scanner);
23         List<Integer> maxInvestments = readIntArray(scanner);
24
25         int maxReturn = 0;
26         int[] bestInvestments = new int[m];
27
28         for (int i = 0; i < m; i++) {
29             if (risks.get(i) > X) continue;
30
31
32             int investI = Math.min(N, maxInvestments.get(i));
33             int retI = investI * returns.get(i);
34             if (retI > maxReturn) {
35                 maxReturn = retI;
36                 Arrays.fill(bestInvestments, 0);
37                 bestInvestments[i] = investI;
38             }
39
40
41             for (int j = i + 1; j < m; j++) {
42                 if (risks.get(i) + risks.get(j) > X) continue;
43
44                 int investJ;
45                 if (returns.get(i) > returns.get(j)) {
```

```

46         investI = Math.min(N, maxInvestments.get(i));
47         investJ = Math.min(N - investI, maxInvestments.get
48     (j));
49     } else {
50         investJ = Math.min(N, maxInvestments.get(j));
51         investI = Math.min(N - investJ, maxInvestments.get
52     (i));
53     }
54     int retPair = investI * returns.get(i) + investJ * return
55     s.get(j);
56     if (retPair > maxReturn) {
57         maxReturn = retPair;
58         Arrays.fill(bestInvestments, 0);
59         bestInvestments[i] = investI;
60         bestInvestments[j] = investJ;
61     }
62 }
63
64 for (int investment : bestInvestments) {
65     System.out.print(investment + " ");
66 }
67 System.out.println();
68 }
69 }

```

## Python

```
1 import sys
2
3 def read_int_array():
4     """ 从标准输入读取一行并转换为整数列表 """
5     return list(map(int, sys.stdin.readline().split()))
6
7 def main():
8
9     m, N, X = read_int_array()
10    returns = read_int_array()
11    risks = read_int_array()
12    max_investments = read_int_array()
13
14    max_return = 0
15    best_investments = [0] * m
16
17    for i in range(m):
18        if risks[i] > X:
19            continue
20
21
22        invest_i = min(N, max_investments[i])
23        ret_i = invest_i * returns[i]
24        if ret_i > max_return:
25            max_return = ret_i
26            best_investments = [0] * m
27            best_investments[i] = invest_i
28
29
30        for j in range(i + 1, m):
31            if risks[i] + risks[j] > X:
32                continue
33
34            if returns[i] > returns[j]:
35                invest_i = min(N, max_investments[i])
36                invest_j = min(N - invest_i, max_investments[j])
37            else:
38                invest_j = min(N, max_investments[j])
39                invest_i = min(N - invest_j, max_investments[i])
40
41            ret_pair = invest_i * returns[i] + invest_j * returns[j]
42            if ret_pair > max_return:
43                max_return = ret_pair
44                best_investments = [0] * m
45                best_investments[i] = invest_i
```



```
46         best_investments[j] = invest_j
47
48     print(" ".join(map(str, best_investments)))
49
50 if __name__ == "__main__":
51     main()
```

## JavaScript

```
1  const readline = require('readline');
2
3  const rl = readline.createInterface({
4    input: process.stdin,
5    output: process.stdout
6  });
7
8  let inputLines = [];
9
10 rl.on('line', (line) => {
11   inputLines.push(line.trim());
12 }).on('close', () => {
13   const readIntArray = (index) => inputLines[index].split(" ").map(Number);
14
15
16   const [m, N, X] = readIntArray(0);
17   const returns = readIntArray(1);
18   const risks = readIntArray(2);
19   const maxInvestments = readIntArray(3);
20
21   let maxReturn = 0;
22   let bestInvestments = new Array(m).fill(0);
23
24   for (let i = 0; i < m; i++) {
25     if (risks[i] > X) continue;
26
27     let investI = Math.min(N, maxInvestments[i]);
28     let retI = investI * returns[i];
29     if (retI > maxReturn) {
30       maxReturn = retI;
31       bestInvestments.fill(0);
32       bestInvestments[i] = investI;
33     }
34
35     for (let j = i + 1; j < m; j++) {
36       if (risks[i] + risks[j] > X) continue;
37
38       let investI, investJ;
39       if (returns[i] > returns[j]) {
40         investI = Math.min(N, maxInvestments[i]);
41         investJ = Math.min(N - investI, maxInvestments[j]);
42       } else {
43         investJ = Math.min(N, maxInvestments[j]);
44         investI = Math.min(N - investJ, maxInvestments[i]);
```

```
45         }
46
47         let retPair = investI * returns[i] + investJ * returns[j];
48         if (retPair > maxReturn) {
49             maxReturn = retPair;
50             bestInvestments.fill(0);
51             bestInvestments[i] = investI;
52             bestInvestments[j] = investJ;
53         }
54     }
55 }
56
57 console.log(bestInvestments.join(" "));
58 });
```

**Go**

```
1  package main
2
3  import (
4      "bufio"
5      "fmt"
6      "os"
7      "strconv"
8      "strings"
9  )
10
11
12  func readIntArray(scanner *bufio.Scanner) []int {
13      scanner.Scan()
14      fields := strings.Fields(scanner.Text())
15      numbers := make([]int, len(fields))
16      for i, field := range fields {
17          numbers[i], _ = strconv.Atoi(field)
18      }
19      return numbers
20  }
21
22  func min(a, b int) int {
23      if a < b {
24          return a
25      }
26      return b
27  }
28
29  func main() {
30      scanner := bufio.NewScanner(os.Stdin)
31
32
33      input := readIntArray(scanner)
34      m, N, X := input[0], input[1], input[2]
35
36
37      returns := readIntArray(scanner)
38      risks := readIntArray(scanner)
39      maxInvestments := readIntArray(scanner)
40
41      maxReturn := 0
42      bestInvestments := make([]int, m)
43
44
45      for i := 0; i < m; i++ {
```

```

46     if risks[i] > X {
47         continue
48     }
49
50
51     investI := min(N, maxInvestments[i])
52     retI := investI * returns[i]
53     if retI > maxReturn {
54         maxReturn = retI
55         for k := range bestInvestments {
56             bestInvestments[k] = 0
57         }
58         bestInvestments[i] = investI
59     }
60
61
62     for j := i + 1; j < m; j++ {
63         if risks[i]+risks[j] > X {
64             continue
65         }
66
67         var investI, investJ int
68
69         if returns[i] > returns[j] {
70             investI = min(N, maxInvestments[i])
71             investJ = min(N-investI, maxInvestments[j])
72         } else {
73             investJ = min(N, maxInvestments[j])
74             investI = min(N-investJ, maxInvestments[i])
75         }
76
77         retPair := investI*returns[i] + investJ*returns[j]
78         if retPair > maxReturn {
79             maxReturn = retPair
80             for k := range bestInvestments {
81                 bestInvestments[k] = 0
82             }
83             bestInvestments[i] = investI
84             bestInvestments[j] = investJ
85         }
86     }
87 }
88
89
90 for i, val := range bestInvestments {
91     if i > 0 {
92         fmt.Print(" ")
93     }

```

```
94         fmt.Print(val)
95     }
96     fmt.Println()
97 }
```

来自: [华为OD机试 2025 B卷 – 最佳投资方式 \(C++ & Python & JAVA & JS & GO\)](#)–CSDN博客

# 华为OD 机考 2025B卷 - 计算网络信号 / 信号强度 (C++ & Python & JAV

## 计算网络信号 / 信号强度

华为OD机试真题目录点击查看: [华为OD机试2025B卷真题题库目录 | 机考题库 + 算法考点详解](#)

华为OD机试2025B卷 100分题型

### 题目描述

网络信号经过传递会逐层衰减，且遇到阻隔物无法直接穿透，在此情况下需要计算某个位置的网络信号值。

注意:网络信号可以绕过阻隔物。

- `array[m][n]` 的二维数组代表网格地图，
- `array[i][j] = 0`代表*i*行*j*列是空旷位置；
- `array[i][j] = x`(*x*为正整数)代表*i*行*j*列是信号源，信号强度是*x*；
- `array[i][j] = -1`代表*i*行*j*列是阻隔物。
- 信号源只有1个，阻隔物可能有0个或多个
- 网络信号衰减是上下左右相邻的网格衰减1

现要求输出对应位置的网络信号值

### 输入描述

输入为三行，

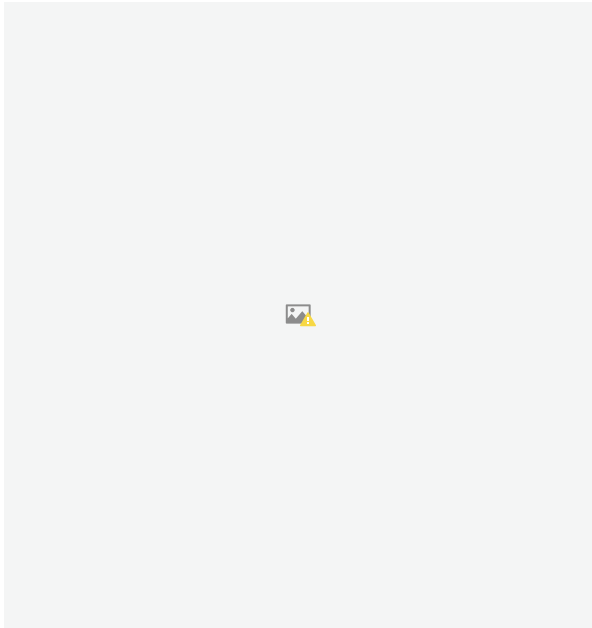
- 第一行为 *m* 、 *n* ， 代表输入是一个 *m* × *n* 的数组。
- 第二行是一串 *m* × *n* 个用空格分隔的整数。每连续 *n* 个数代表一行，再往后 *n* 个代表下一行，以此类推。对应的值代表对应的网格是空旷位置，还是信号源，还是阻隔物。
- 第三行是 *i* 、 *j*，代表需要计算`array[i][j]`的网络信号值。

注意：此处 *i* 和 *j* 均从 0 开始，即第一行 *i* 为 0。

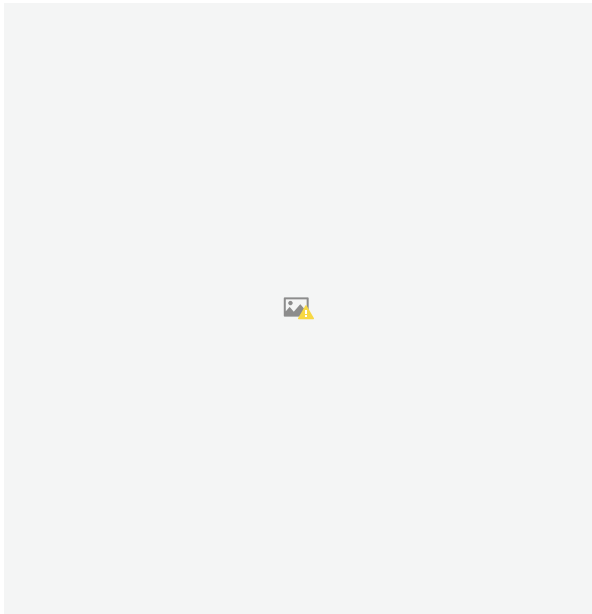
▼ Plain Text

```
1 6 5
2 0 0 0 -1 0 0 0 0 0 0 0 0 -1 4 0 0 0 0 0 0 0 0 0 -1 0 0 0 0 0
3 1 4
```

代表如下地图：



需要输出第1行第4列的网络信号值，值为2。



## 输出描述

输出对应位置的网络信号值，如果网络信号未覆盖到，也输出0。  
一个网格如果可以途径不同的传播衰减路径传达，取较大的值作为其信号值。

## 示例1

### 输入



▼ Plain Text |

```
1 6 5
2 0 0 0 -1 0 0 0 0 0 0 0 0 0 -1 4 0 0 0 0 0 0 0 0 0 0 -1 0 0 0 0 0
3 1 4
```

输出

## 示例2

输入

▼ Plain Text |

```
1 6 5
2 0 0 0 -1 0 0 0 0 0 0 0 0 0 -1 4 0 0 0 0 0 0 0 0 0 0 0 0 0 0
3 2 1
```

输出

## 题解

思路： BFS

- 1. 一道经典的多源 BFS 模板题。
- 2. 使用 队列 模拟进行 BFS 扩散。初始将所有值大于 0 的位置加入到队列中。接下来循环迭代将队列中的元素进行四周扩散，更新四周的信号值。具体逻辑参照下面代码
- 3. 输出指定位置的信号强度。



```
1  #include <iostream>
2  #include <vector>
3  #include <queue>
4
5  using namespace std;
6
7  int main() {
8      int rows, cols;
9      cin >> rows >> cols;
10
11     vector<int> grid(rows * cols);
12     queue<pair<int, int>> bfsQueue;
13
14
15     for (int i = 0; i < rows; i++) {
16         for (int j = 0; j < cols; j++) {
17             cin >> grid[i * cols + j];
18             if (grid[i * cols + j] > 0) {
19                 bfsQueue.emplace(i, j);
20             }
21         }
22     }
23
24
25     constexpr pair<int, int> directions[] = {{-1, 0}, {1, 0}, {0, -1},
26 {0, 1}};
27
28     while (!bfsQueue.empty()) {
29         auto [x, y] = bfsQueue.front();
30         bfsQueue.pop();
31
32         int currentSignal = grid[x * cols + y];
33         if (currentSignal == 1) continue;
34
35         for (auto [dx, dy] : directions) {
36             int newX = x + dx, newY = y + dy;
37             if (newX >= 0 && newX < rows && newY >= 0 && newY < cols && gr
id[newX * cols + newY] == 0) {
38                 grid[newX * cols + newY] = currentSignal - 1;
39                 bfsQueue.emplace(newX, newY);
40             }
41         }
42     }
43
```

```
44     int targetX, targetY;  
45     cin >> targetX >> targetY;  
46     cout << grid[targetX * cols + targetY] << endl;  
47  
48     return 0;  
49 }
```

## Java

```
1  import java.util.*;
2
3  public class Main {
4      public static void main(String[] args) {
5          Scanner scanner = new Scanner(System.in);
6          int rows = scanner.nextInt();
7          int cols = scanner.nextInt();
8
9          int[] grid = new int[rows * cols];
10         Queue<int[]> bfsQueue = new LinkedList<>();
11
12
13         for (int i = 0; i < rows; i++) {
14             for (int j = 0; j < cols; j++) {
15                 int index = i * cols + j;
16                 grid[index] = scanner.nextInt();
17                 if (grid[index] > 0) {
18                     bfsQueue.offer(new int[]{i, j});
19                 }
20             }
21         }
22
23
24         int[][] directions = {{-1, 0}, {1, 0}, {0, -1}, {0, 1}};
25
26
27         while (!bfsQueue.isEmpty()) {
28             int[] pos = bfsQueue.poll();
29             int x = pos[0], y = pos[1];
30             int currentSignal = grid[x * cols + y];
31
32             if (currentSignal == 1) continue;
33
34             for (int[] dir : directions) {
35                 int newX = x + dir[0], newY = y + dir[1];
36                 if (newX >= 0 && newX < rows && newY >= 0 && newY < cols &
37 & grid[newX * cols + newY] == 0) {
38                     grid[newX * cols + newY] = currentSignal - 1;
39                     bfsQueue.offer(new int[]{newX, newY});
40                 }
41             }
42
43             int targetX = scanner.nextInt();
44             int targetY = scanner.nextInt();
```

```
45         System.out.println(grid[targetX * cols + targetY]);
46     }
47 }
```

## Python

```
▼ Plain Text |
1  import sys
2  from collections import deque
3
4
5  rows, cols = map(int, sys.stdin.readline().split())
6
7
8  data = list(map(int, sys.stdin.readline().split()))
9  grid = data[:]
10 bfs_queue = deque()
11
12 for i in range(rows * cols):
13     if grid[i] > 0:
14         bfs_queue.append((i // cols, i % cols))
15
16
17 directions = [(-1, 0), (1, 0), (0, -1), (0, 1)]
18
19
20 while bfs_queue:
21     x, y = bfs_queue.popleft()
22     current_signal = grid[x * cols + y]
23
24     if current_signal == 1:
25         continue
26
27     for dx, dy in directions:
28         new_x, new_y = x + dx, y + dy
29         if 0 <= new_x < rows and 0 <= new_y < cols and grid[new_x * cols
+ new_y] == 0:
30             grid[new_x * cols + new_y] = current_signal - 1
31             bfs_queue.append((new_x, new_y))
32
33
34 target_x, target_y = map(int, sys.stdin.readline().split())
35 print(grid[target_x * cols + target_y])
```

## JavaScript

```
1  const readline = require("readline");
2
3  const rl = readline.createInterface({
4    input: process.stdin,
5    output: process.stdout
6  });
7
8  let inputLines = [];
9  rl.on("line", (line) => {
10    inputLines.push(line);
11  }).on("close", () => {
12    let [rows, cols] = inputLines[0].split(" ").map(Number);
13    let grid = inputLines[1].split(" ").map(Number);
14    let bfsQueue = [];
15
16    for (let i = 0; i < grid.length; i++) {
17      if (grid[i] > 0) {
18        bfsQueue.push([Math.floor(i / cols), i % cols]);
19      }
20    }
21
22
23    const directions = [[-1, 0], [1, 0], [0, -1], [0, 1]];
24
25
26    while (bfsQueue.length) {
27      let [x, y] = bfsQueue.shift();
28      let currentSignal = grid[x * cols + y];
29
30      if (currentSignal === 1) continue;
31
32      for (let [dx, dy] of directions) {
33        let newX = x + dx, newY = y + dy;
34        if (newX >= 0 && newX < rows && newY >= 0 && newY < cols && grid[newX * cols + newY] === 0) {
35          grid[newX * cols + newY] = currentSignal - 1;
36          bfsQueue.push([newX, newY]);
37        }
38      }
39    }
40
41    let [targetX, targetY] = inputLines[2].split(" ").map(Number);
42    console.log(grid[targetX * cols + targetY]);
43  });
```



Go

```
1 package main
2
3 import (
4     "bufio"
5     "fmt"
6     "os"
7     "strconv"
8     "strings"
9 )
10
11 func main() {
12     scanner := bufio.NewScanner(os.Stdin)
13     scanner.Scan()
14     dimensions := strings.Fields(scanner.Text())
15     rows, _ := strconv.Atoi(dimensions[0])
16     cols, _ := strconv.Atoi(dimensions[1])
17
18     scanner.Scan()
19     data := strings.Fields(scanner.Text())
20     grid := make([]int, rows*cols)
21     var bfsQueue [][]int
22
23     for i := 0; i < rows*cols; i++ {
24         grid[i], _ = strconv.Atoi(data[i])
25         if grid[i] > 0 {
26             bfsQueue = append(bfsQueue, [2]int{i / cols, i % cols})
27         }
28     }
29
30
31     directions := [][]int{{-1, 0}, {1, 0}, {0, -1}, {0, 1}}
32
33
34     for len(bfsQueue) > 0 {
35         x, y := bfsQueue[0][0], bfsQueue[0][1]
36         bfsQueue = bfsQueue[1:]
37
38         if grid[x*cols+y] == 1 {
39             continue
40         }
41
42         for _, d := range directions {
43             newX, newY := x+d[0], y+d[1]
44             if newX >= 0 && newX < rows && newY >= 0 && newY < cols && grid[newX*cols+newY] == 0 {
```

```

45         grid[newX*cols+newY] = grid[x*cols+y] - 1
46         bfsQueue = append(bfsQueue, [2]int{newX, newY})
47     }
48 }
49 }
50 }
51 scanner.Scan()
52 target := strings.Fields(scanner.Text())
53 targetX, _ := strconv.Atoi(target[0])
54 targetY, _ := strconv.Atoi(target[1])
55 fmt.Println(grid[targetX*cols+targetY])
56 }

```

来自: [华为OD 机考 2025B卷 - 计算网络信号 / 信号强度 \(C++ & Python & JAVA & JS & GO\)-CSDN博客](#)