算法分析习题选讲(第二章)

chyx111@qq.com

1150 1151 1515 魔板 题目大意

给出魔板的起始状态，三种基本操作，步数上限和目标状态， 求从起始状态到目标状态的操作序列，长度不得超过上限。

1150 1151 1515 魔板 三种基本操作

- A:上下互换

1	2	3	4
8	7	6	5

8	7	6	5
1	2	3	4

- B:循环右移

1	2	3	4
8	7	6	5

4	1	2	3
5	8	7	6

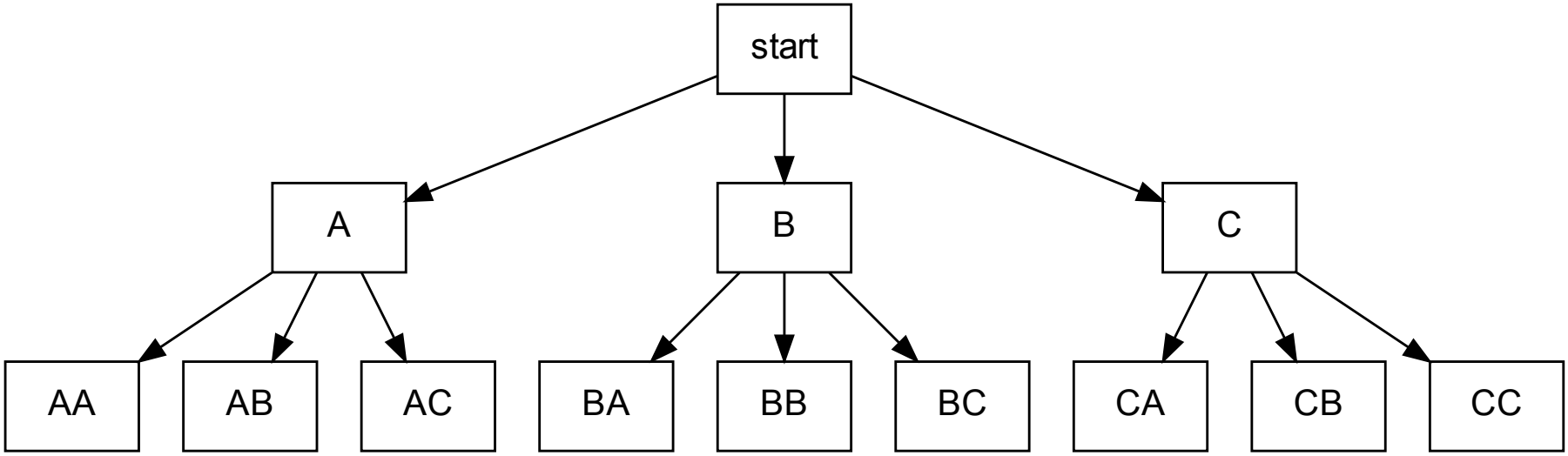
- C:中间四块顺时针转

1	2	3	4
8	7	6	5

1	7	2	4
8	6	3	5

1150 1151 1515 魔板 解题思路

- 对模板进行状态搜索
- 由一种状态可以转移到另外三种状态，搜索树为一棵三叉树
- 在这棵三叉树上搜索，目的是求出最优解。



三叉树

1150 1151 1515 魔板 算法一：DFS(深度优先搜索)

- 对这棵三叉树进行DFS
- 缺点：若想求得最优解，需要遍历整棵树，会遍历很多重复的节点
- 优化：若已找到一个可行解，可剪去大于等于这个深度的所有子树
- 效果：勉强可过1150
- 评价：很傻很天真

1150 1151 1515 魔板 算法二：BFS(广度优先搜索)

- 对这棵三叉树进行BFS
- 第一个可行解即是最优解
- 效果：轻松切掉1150，但过不了1151
- 评价：很慢很暴力

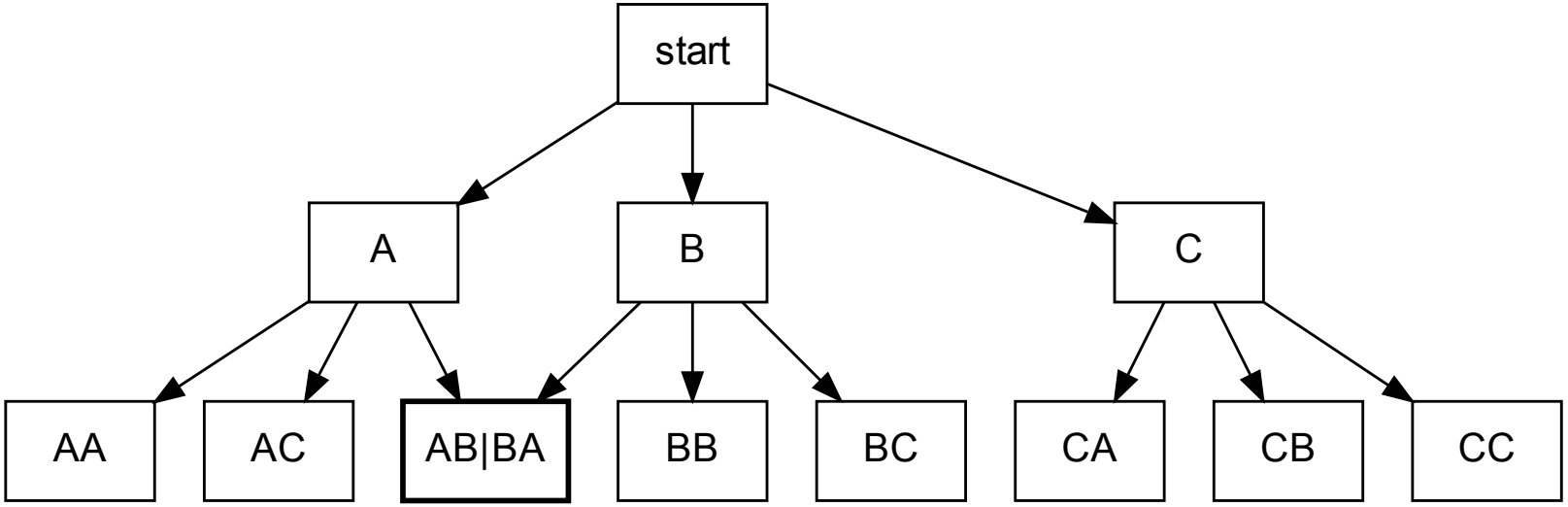
1150 1151 1515 魔板 算法三：BFS + 判重

- 对这棵三叉树进行BFS
- 第一个可行解即是最优解
- 判重

BFS每经过一个节点，就把它放进已搜索列表中

如果节点在已搜索列表存在，则不再扩展该节点

共有8!=40320个节点



三叉树

1150 1151 1515 魔板 算法三：BFS + 判重

节点编码

1. 康托展开
2. 自定义编码，如把状态

1	2	3	4
8	7	6	5

编码为整数12348765

1150 1151 1515 魔板 *康托展开

康托展开是一个全排列到一个自然数的双射，常用于构建哈希表时的空间压缩

N位的十进制整数可以由N个< 10的数字表示

$$X = a_n 10^{n-1} + a_{n-1} 10^{n-2} + \ldots + a_i 10^{i-1} + \ldots + a_1 10^0$$

类似的，N个数字的排列可以由N个$< N$的数字表示

$$X = a_n (n-1)! + a_{n-1} (n-2)! + \ldots + a_i (i-1)! + \ldots + a_1 0!$$

a_i表示，在全排列π中，比π_i大而且位于π_i前面的数字的个数。

1150 1151 1515 魔板 代码

```
int const kMaxState = 40320;
bool visit[kMaxState];  // 8!
int parent[kMaxState];
int operation[kMaxState];

queue<State> que;
void update(const State& state, const State& new_state, char op) {
  que.push(new_state);
  visit[new_state.hash_value] = true;
  parent[new_state.hash_value] = state.hash_value;
  operation[new_state.hash_value] = op;
}

void bfs(State start) {
  memset(visit, 0, sizeof (visit));
  que.push(start);
  visit[start.hash_value] = true;
  while (!que.empty()) {
    State state = que.front();
    que.pop();

    State new_state = state; new_state.transformA();
    if (!visit[new_state.hash_value]) update(state, new_state, 'A');

    new_state = state; new_state.transformB();
    if (!visit[new_state.hash_value]) update(state, new_state, 'B');

    new_state = state; new_state.transformC();
    if (!visit[new_state.hash_value]) update(state, new_state, 'C');
  }
}
```

1007 To and Fro 题目大意

给出一种加密方式，把一个字符串按列写成二维形式，再逐行从左到右或从右到左交替输出

给出加密后的字符串，还原本来的字符串。

```
theresnoplacelikehomeonasnowynightx

toioy
hpknn
eleai
rahsg
econh
semot
nlewx

toioynnkpheleaigshareconhtomesnlewx
```

1007 To and Fro 解题思路

按照解密规则把输入字符串写在二维数组上，再输出。

```
int pos = 0;
for (int i = 0; i < n; i++) {
  for (int j = 0; j < m; j++) {
    if(i % 2 != 0) {
      ans[i][m - 1 - j] = enc[pos];
    } else {
      ans[i][j] = enc[pos];
    }
    pos++;
  }
}
```

1036 Crypto Columns 题目大意

加密方式：

- 给定一个Keyword
- 把一个字符串按行写成二维形式
- 按照Keyword的字符大小顺序逐列输出
- 给出加密后的字符串，还原本来的字符串。

```
BATBOY
MEETMEBYTHEOLDOAKTREENTH

MEETME
BYTHEO
LDOAKT
REENTH

B A T B O Y
M E E T M E
B Y T H E O
L D O A K T
R E E N T H

EYDEMBLRTHANMEKTETOEEOTH
```

1036 Crypto Columns 解题思路

按照解密规则把输入字符串写在二维数组上，再输出。

```
int m = keyword.size();
vector<pair<char, int> > order;
for (int i = 0; i < m; ++i) {
  order.push_back(make_pair(keyword[i], i));
}
sort(order.begin(), order.end());  //得到keyword的字母顺序
int n = enc.size() / m;
vector<vector<char> > matrix(n, vector<char>(m, 0));
int pos = 0;
for (int j = 0; j < m; ++j) {
  for (int i = 0; i < n; ++i) {
    matrix[i][order[j].second] = enc[pos];
    ++pos;
  }
}
```

1006 Team Rankings 题目大意

对于两个排列π_1，π_2，定义 distance(π_1, π_2)为在π_1，π_2中出现的相对次序不同的元素的对数

相当于以π_1为基准，求π_2的逆序数。

如：π_1 = 2, 1, 3, 4

π_2 = 1, 3, 2, 4

不同的对有：(1,2), (2,3)，所以距离为2

如果以π_1为基准调整π_2，则π_2变为：

2, 3, 1, 4

它的逆序数也为2

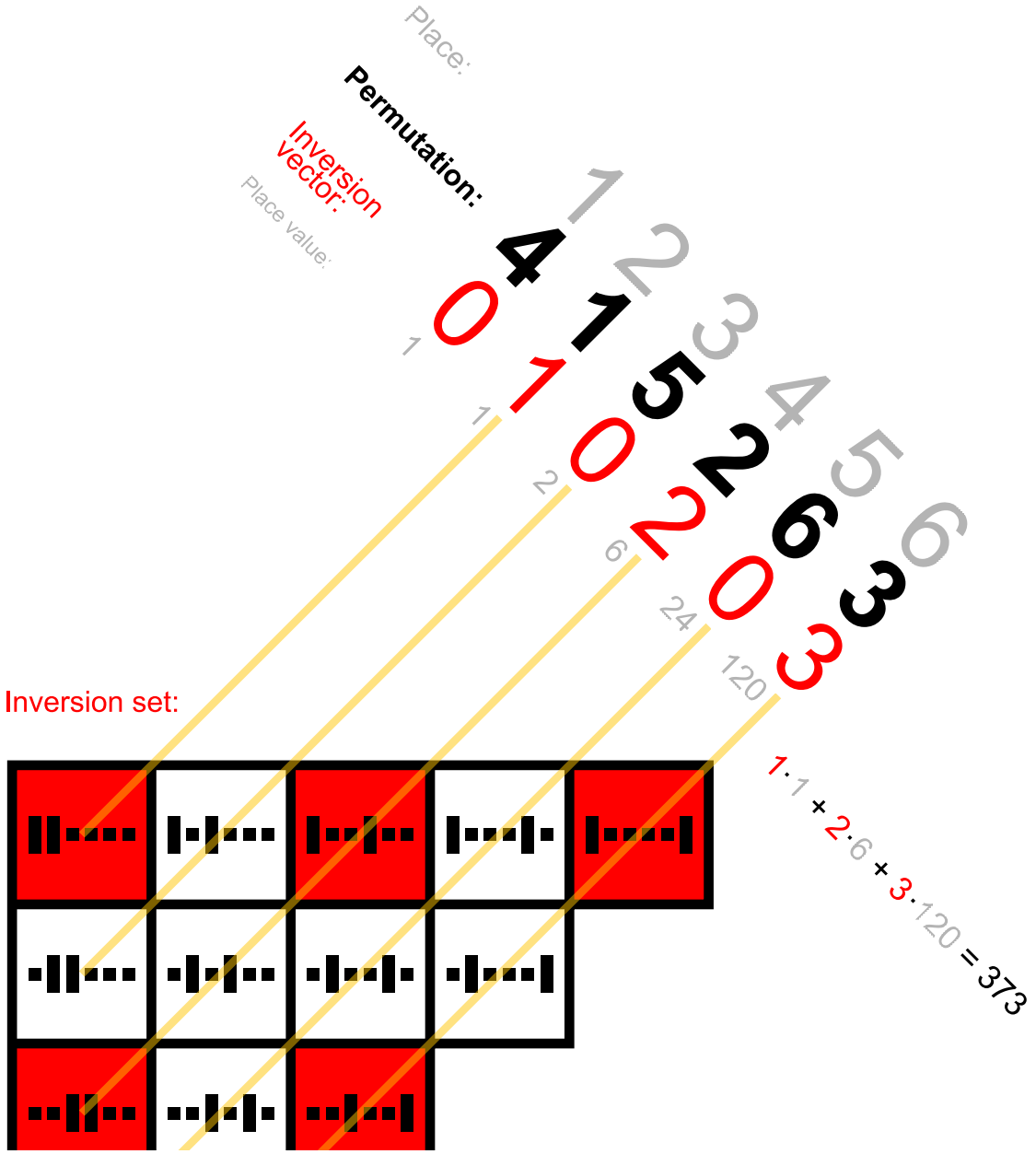
1006 Team Rankings 题目大意

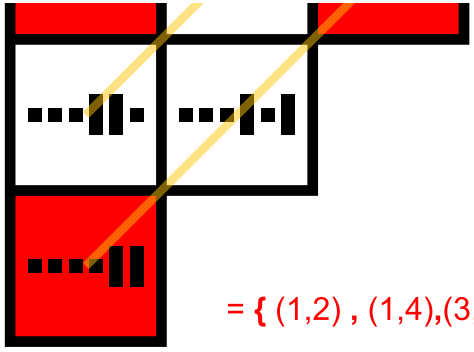
对于两个排列π_1，π_2，定义 distance(π_1, π_2)为在π_1，π_2中出现的相对次序不同的元素的对数

相当于以π_1为基准，求π_2的逆序数

现在给出n($n \leq 100$)个5元排列，要求构造一个排列，使得该排列对n个排列的distance之和最小

1006 Team Rankings 逆序数





= { (1,2) , (1,4),(3,4) , (1,6),(3,6),(5,6) }

1006 Team Rankings 解题思路

- 可以使用深度(或宽度)优先法生成排列。
- 求逆序数的算法
- 平方级枚举 $O(n^2)$
- 规模较大时可采用归并排序 $O(n \log n)$

1006 Team Rankings 代码

```
int const kMaxDigit = 5;
bool visit[kMaxDigit];

int get_distance(string sa, string sb) {
  static int posa[128], posb[128];
  for (int i = 0; i < kMaxDigit; ++i) {
    posa[sa[i]] = i;
    posb[sb[i]] = i;
  }
  int inv = 0;
  for (char c1 = 'A'; c1 <= 'E'; ++c1) {
    for (char c2 = c1 + 1; c2 <= 'E'; ++c2) {  //不同的对
      if ((posa[c1] - posa[c2]) * (posb[c1] - posb[c2]) < 0) {
        inv++;
      }
    }
  }
  return inv;
}

string perm;
pair<int, string> best_perm;
void Solve() {
  int sum = 0;
  for (int i = 0; i < n; ++i) {
    sum += get_distance(perm, ranks[i]);
  }
  best_perm = min(best_perm, make_pair(sum, perm));
}


void dfs(int pos) {
  if (pos == kMaxDigit) {
    Solve();
    return;
  }
  for (int x = 0; x < kMaxDigit; ++x) if (!visit[x]){
```

```
            perm[pos] = 'A' + x;
            visit[x] = true;
            dfs(pos + 1);
            visit[x] = false;
        }
    }
```

1009 Mersenne Composite N 题目大意

- 梅森素数M_p：形如2^p-1的素数，其中p也必须为素数
- 给定k，求出所有素数$n \leq k$，满足条件：2^n-1不是梅森素数，并要求对这些形如2^n-1的数字进行因子分解。
- $k \leq 63$

1009 Mersenne Composite N 解题思路

方法一：

通过网络查找梅森素数的性质：

对M_q（q是素数）有：

若a是Mq的因数，则a有如下性质：

- $a \equiv 1 \pmod{2q}$
- $a \equiv \pm 1 \pmod{8}$

对每个数，枚举所有可能的因数，测试是否能分解。

方法二：

查找资料可知在$n \leq 63$内有以下M_n满足答案

11,23,29,37,41,43,47,53,59

只对这些数进行分解即可。

1009 Mersenne Composite N 代码

```
vector<pair<long long, int> > factor(long long x) {
  vector<pair<long long, int> > factor;
  long long n = (1LL << x) - 1;
  for (int i = 3; i * i <= n; i += 2) {
    long long cnt = 0;
    while (n % i == 0) {
      n /= i;
      ++cnt;
    }
    if (cnt != 0) factor.push_back(make_pair(i, cnt));
  }
  return factor;
}
```

1050 Numbers & Letters 题目大意

给出5个数和一个目标数，从5个数中选出一部分数通过加减乘除运算得到小于等于目标数的最大数。

类似的题目：24点，从52张牌中抽4张，使得其加减乘除得24

1050 Numbers & Letters 解题思路

DFS求出所有可能的运算组合和顺序，得到最接近目标数的答案。

1050 Numbers & Letters 代码

```
void dfs(const vector<int>& arr) {
  if (arr.size() <= 1) return;

  for (int i = 0; i < arr.size(); ++i) {
    for (int j = i + 1; j < arr.size(); ++j) {
      vector<int> new_arr;
      for (int k = 0; k < arr.size(); ++k) if (i != k && j != k) {
        new_arr.push_back(arr[k]);
      }
      new_arr.push_back(-1); // to be modified

      new_arr.back() = arr[i] + arr[j]; update(new_arr.back()); dfs(new_arr);
      new_arr.back() = abs(arr[i] - arr[j]); update(new_arr.back()); dfs(new_arr);
      new_arr.back() = arr[i] * arr[j]; update(new_arr.back()); dfs(new_arr);

      if (arr[j] != 0 && arr[i] % arr[j] == 0) {
        new_arr.back() = arr[i] / arr[j]; update(new_arr.back()); dfs(new_arr);
      }
      if (arr[i] != 0 && arr[j] % arr[i] == 0) {
        new_arr.back() = arr[j] / arr[i]; update(new_arr.back()); dfs(new_arr);
      }
    }
  }
}
```

陷阱：#3注意目标数字的范围。

1443 Printer Queue 题目大意

有一个长度为n的打印任务队列，每个任务有优先级

每次从队列头得到一个任务，如果它是剩余任务中优先级最高的，则打印它，否则放到队列尾

问其中某个任务是第几个被打印的。

$n \leq 100$

1443 Printer Queue 解题思路

使用队列直接模拟

取出队列头判断是否打印，如果打印则已打印任务数加一

直到特定的任务完成，输出答案

1443 Printer Queue 代码

```
map<int, int> cnt_priority;
for (int i = 0; i < n; ++i) {
  cnt_priority[priority[i]]++;
}
int minute = 0;
for (; ;) {
  int highest = cnt_priority.rbegin()->first;
  while (priority[que.front()] != highest) {
    que.push(que.front());
    que.pop();
  }
  ++minute;
  if (que.front() == target) {
    break;
  }
  if (--cnt_priority[highest] == 0) {
    cnt_priority.erase(highest);
  }
  que.pop();
}
```

1156 Binary tree 题目大意

```
3
4 C 1 3
1 A 0 0
3 B 0 0
```

给出一棵二叉树每个节点的编号，内容以及左右子节点的编号

要求对二叉树进行先序遍历，输出每个节点的内容。

1156 Binary tree 解题思路

- 先序遍历：先输出当前节点的内容，然后遍历左子树，最后遍历右子树。
- 先找出没有父节点的节点，即根。
- 从根开始遍历进行先序遍历。

```
void Preorder(Node *s) {
  Visit(s);
  Preorder(s->leftchild);
  Preorder(s->rightchild);
}
```

1063 Who's the Boss 题目大意

有n个人的编号，身高和工资

一个人的直接上司是身高不比他小且工资比他高最少的人

而一个的下属不止是他的直接下属，还包含他的下属的下属，等等

现在有q个询问，你需要输出询问到的人的直接上司，以及他的下属的数量

1063 Who's the Boss 解题思路

人数较多≤ 30000

必须构思比$O(n^2)$快的算法

排序 两个变量：身高，工资

1063 Who's the Boss 解法一

先按工资排序

假设排完序后的身高依次为：

工资 **25 20 15 10 5**

身高 180 160 170 165 175

工资 **25 20 15 10 5**

身高 180

身高 **180** *160*

身高 **180** 160 *170*

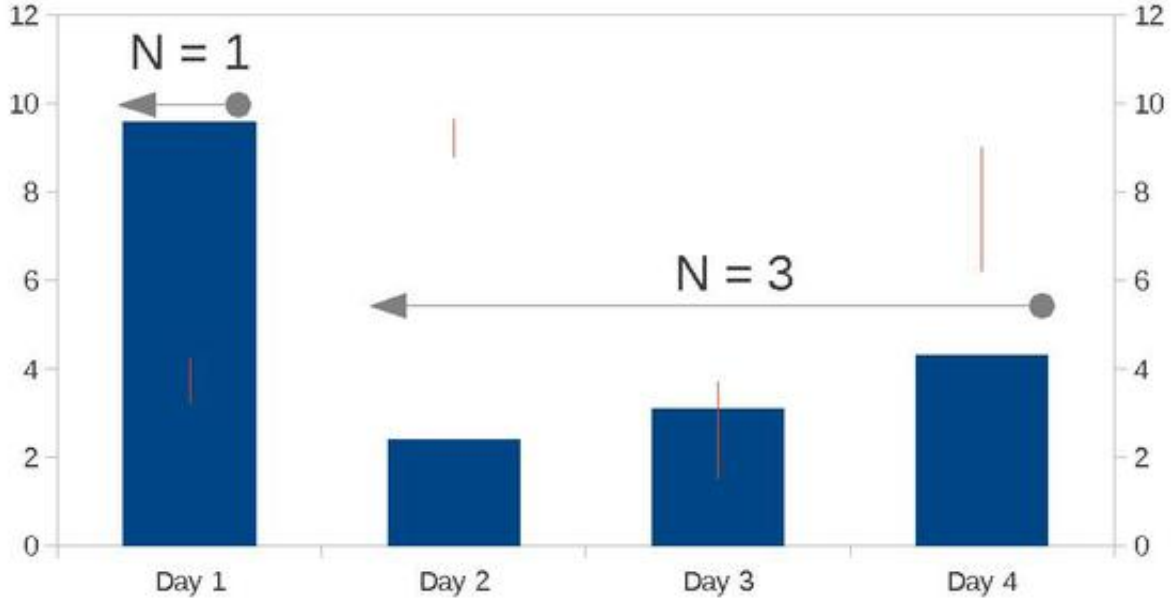
身高 180 160 **170** *165*

身高 **180** 160 170 165 *175*

1063 Who's the Boss 解法一

维护一个单调递减的栈

http://en.wikipedia.org/wiki/Stack_(abstract_data_type)#The_Stock_Span_Problem



1063 Who's the Boss 解法二

先按身高排序

假设排完序后的身高依次为：

工资 **25 5 15 10 20**

身高 180 175 170 165 160

身高 **180 175 170 165 160**

工资 25

工资 **25** *5*

工资 **25** 5 *15*

工资 25 5 **15** *10*

工资 **25** 5 15 10 *20*

1063 Who's the Boss 解法二

用set查找集合中比某个元素大的最小的元素

upper_bound

1063 Who's the Boss 代码

```
struct employee {
  int height, id, earn, number, manager, sub;
};
bool cmp(const employee &a,const employee &b) {
  if (a.height != b.height) {
    return a.height < b.height;
  } else {
    return a.earn < b.earn;
  }
}
bool cmp2(const employee &a,const employee &b) {
    return a.id<b.id;
}
set<employee> h;
```

```
sort(a, a + n, cmp);
for (int i = 0; i < n; i++) {
  a[i].number = i;
}
for (int i = n - 1; i >= 0; i--) {
  set<employee>::iterator it = h.upper_bound(a[i]);
  if (it == h.end()) {
    a[i].manager = -1;
  } else {
    a[i].manager = it->number;
  }
  h.insert(a[i]);
}
for (int i = 0; i < n; i++) {
  if (a[i].manager != -1) {
    a[a[i].manager].sub += a[i].sub + 1;
    a[i].manager = a[a[i].manager].id;
  } else {
    a[i].manager = 0;
  }
}
```

1024 Magic Island 解题思路

两个节点不经过重复边的路径有且只有一条

树

问从某个点出发最远可以走多远

1024 Magic Island 解法

从该点开始做DFS或BFS，取最大路径长度即可

树的DFS的一般写法：

```
struct Node {
  int to, length;
};
vector<Node> G[kMaxNumNode];
void dfs(int x, int parent, int length) {
  ans = max(length, ans);
  for (int i = 0; i < G[x].size(); ++i) {
    if (G[x][i].to != parent) {
      dfs(G[x][i], x, length + G[x][i].length);
    }
  }
}
```