**★★★★☆**

**題組：Problem Set Archive** **with Online Judge**

**題號：11041: Quarter-Finals with Brazil!? No!!!**

**解題者：盧品言**

**解題日期：2025年5月8日**

**題意：**

在單敗淘汰賽中，2^N強賽需要進行N輪比賽，例如:8強賽需要3輪比賽，第1輪後剩4隊，第2輪後剩2隊，第3輪後剩1隊。給定一個N值，隊伍命名規則為A,B,C,…X,Y,Z,a,b,c,…x,y,z，每隊都有一個戰鬥力數值，記做s(隊伍名稱)，隊伍A對隊伍B獲勝的機率為s(A)/(s(A)+s(B))，而每一輪第1隊會跟第2隊進行對戰，第3隊會跟第4隊進行對戰，以此類推。給出自己的隊伍名稱，求最佳的賽程排序，使得自己的隊伍最容易獲勝。如果有多種最佳的賽程排序，印出字母序最小的。

**題意範例：**

Sample Input

3 (測資數量)

2 (進行2輪比賽，i.e. 4強賽) 🡨第1組測資

A (自己的隊伍名稱為A)

7 3 1 9 (隊伍ABCD的戰鬥力數值分別為7 3 1 9)

2 (進行2輪比賽，i.e. 4強賽) 🡨第2組測資

C (自己的隊伍名稱為C)

44 44 44 44 (隊伍ABCD的戰鬥力數值分別為44 44 44 44)

3 (進行3輪比賽，i.e. 8強賽) 🡨第3組測資

E (自己的隊伍名稱為E)

8 9 8 9 7 8 9 8 (隊伍ABCDEFGH的戰鬥力數值分別為8 9 8 9 7 8 9 8)

Sample Output

ACBD (第1組測資的最佳解)

ABCD (第2組測資的最佳解)

ABDGCEFH (第3組測資的最佳解)

**解法：**

1.將所有隊伍的戰鬥力作排序。
(自己的隊伍排第一位，剩下的隊伍從戰鬥力低的依序排到戰鬥力高的，若遇兩隊戰鬥力相同，則依字母序排列)

2.兩兩對戰。

3.對半分割之後找出兩邊的字母排序最高的字母，將有較優先字母的那一半放到前面。

4.重複步驟3直到無法再分割。

**解法範例：**

Input:

3

E

22213545

Input表示:

隊伍ABCDEFGH的戰鬥力分別為22213545

自己的隊伍為E(其戰鬥力為3)

步驟1:

將自己的隊伍排在第一位，其他隊伍由戰鬥力低依序排到高，若

遇兩隊戰鬥力相同，則依字母序排列

隊伍戰鬥力:31222455

隊伍排序:EDABCGFH

步驟2:

因為兩兩進行對戰，所以

隊伍戰鬥力:31 22 24 55

隊伍排序:ED AB CG FH

步驟3:

對半分割排序

ED AB | CG FH (比較兩分割中最小的)

🡪ED AB | CG FH

🡪ED | AB || CG | FH

🡪AB | ED || CG | FH

🡪A | B || E | D ||| C | G || F | H

🡪A | B || D | E ||| C | G || F | H

最終解答:

ABDECGFH

**討論：**

1.時間複雜度O(nlogn)，因為前半部分演算法為sorting其時間複雜度為O(nlogn)，而後半部分演算法切半的動作要進行logn輪，而每輪要掃描n個值，故複雜度也為O(nlog(n))。

結合前半與後半演算法，得出量級為nlogn。

2.若使用暴力法，x個隊伍共有x!種排列，針對每一種排列都要計算其自己的隊伍贏的機率，再從中選出機率最小的排列，並在所有最小的排列之中再找出字典序最小的。

在計算自己隊伍贏的機率的過程中需要計算大量的條件機率，計算量過大。

**程式：**

#include <cstdio>

#include <cstring>

struct Team

{

 long long score;

 char name;

};

void swap\_char(char &a, char &b)

{

 char t = a; a = b; b = t;

}

// [l..l+len-1] 與 [r..r+len-1]

void swap\_block(char s[], int l, int r, int len)

{

 for (int i = 0; i < len; i++)

 {

 swap\_char(s[l + i], s[r + i]);

 }

}

// 找 s[l..r] 範圍內的最小字母

char find\_min(char s[], int l, int r)

{

 char mn = s[l];

 for (int i = l + 1; i <= r; i++)

 {

 if (s[i] < mn) mn = s[i];

 }

 return mn;

}

// 依題意遞迴對半調整

void dfsArrange(char s[], int l, int r)

{

 if (l >= r) return;

 int m = (l + r) / 2;

 char minL = find\_min(s, l, m);

 char minR = find\_min(s, m+1, r);

 if (minR < minL)

 {

 int len = m - l + 1;

 swap\_block(s, l, m+1, len);

 }

 dfsArrange(s, l, m);

 dfsArrange(s, m+1, r);

}

int main()

{

 int M;

 if (scanf("%d", &M)!=1) return 0;

 while (M--)

 {

 int N;

 char me;

 scanf("%d\n%c", &N, &me);

 int T = 1 << N;

 long long scores[32];

 for (int i = 0; i < T; i++)

 {

 scanf("%lld", &scores[i]);

 }

 // 蒐集 other 隊伍

 Team other[32];

 int oc = 0;

 for (int i = 0; i < T; i++)

 {

 char team = (i < 26 ? 'A' + i : 'a' + (i - 26));

 if (team == me) continue;

 other[oc].score = scores[i];

 other[oc].name = team;

 oc++;

 }

 // 插入排序：score 小→大，score 相同 name 小→大

 for (int i = 1; i < oc; i++)

 {

 Team key = other[i];

 int j = i - 1;

 while (j >= 0 &&

 (other[j].score > key.score ||

 (other[j].score == key.score && other[j].name > key.name)))

 {

 other[j+1] = other[j];

 j--;

 }

 other[j+1] = key;

 }

 // 建立初始 bracket

 static char s[33];

 s[0] = me;

 for (int i = 0; i < oc; i++)

 {

 s[i+1] = other[i].name;

 }

 s[T] = '\0';

 // 遞迴調整

 dfsArrange(s, 0, T-1);

 // 輸出

 printf("%s\n", s);

 }

 return 0;

}