**合并石子**

问题描述

　　在一条直线上有n堆石子，每堆有一定的数量，每次可以将两堆相邻的石子合并，合并后放在两堆的中间位置，合并的费用为两堆石子的总数。求把所有石子合并成一堆的最小花费。

输入格式

　　输入第一行包含一个整数n，表示石子的堆数。  
　　接下来一行，包含n个整数，按顺序给出每堆石子的大小 。

输出格式

　　输出一个整数，表示合并的最小花费。

样例输入

5  
1 2 3 4 5

样例输出

33

数据规模和约定

　　1<=n<=1000, 每堆石子至少1颗，最多10000颗。

**C++:**

#include <cstdio>

#include <cstring>

#include <iostream>

#include <algorithm>

#include <vector>

#include <set>

#include <map>

#include <cmath>

#include <queue>

using namespace std;

template <class T> void checkmin(T &t,T x) {if(x < t) t = x;}

template <class T> void checkmax(T &t,T x) {if(x > t) t = x;}

template <class T> void \_checkmin(T &t,T x) {if(t==-1) t = x; if(x < t) t = x;}

template <class T> void \_checkmax(T &t,T x) {if(t==-1) t = x; if(x > t) t = x;}

typedef pair <int,int> PII;

typedef pair <double,double> PDD;

typedef long long ll;

#define foreach(it,v) for(\_\_typeof((v).begin()) it = (v).begin(); it != (v).end ; it ++)

const int N = 50050;

int n , t , stone[N] , ans;

void combine(int k) {

int tmp = stone[k] + stone[k-1];

ans += tmp;

for(int i=k;i<t-1;i++)

stone[i] = stone[i+1];

t --;

int j;

for(j=k-1;j>0 && stone[j-1]<tmp;j--)

stone[j] = stone[j-1];

stone[j] = tmp;

while(j >= 2 && stone[j] >= stone[j-2]) {

int d = t - j;

combine(j-1);

j = t-d;

}

}

int main() {

while(~scanf("%d",&n) && n) {

for(int i=0;i<n;i++) scanf("%d",&stone[i]);

t = 1 , ans = 0;

for(int i=1;i<n;i++) {

stone[t++] = stone[i];

while(t >= 3 && stone[t-3] <= stone[t-1]) {

combine(t-2);

}

}

while(t > 1) combine(t-1);

printf("%d\n" , ans);

}

return 0;

}

**Java:**

import java.util.Scanner;

public class Main {

static int n;

static int count[][], s[][];

static int value[], sum[];

static int min(int a, int b) {

if (a < b)

return a;

else

return b;

}

public static void main(String[] args) {

// TODO Auto-generated method stub

Scanner reader = new Scanner(System.in);

n = reader.nextInt();

value = new int[n];

sum = new int[n + 1];

s = new int[n][n];

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

value[i] = reader.nextInt();

sum[0] = 0;

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

sum[i] = sum[i - 1] + value[i - 1];

count = new int[n][n];

for (int i = 0; i < n - 1; i++) {

count[i][i + 1] = value[i] + value[i + 1];

s[i][i + 1] = i;

}

for (int i = n - 3; i >= 0; i--) {

for (int j = i + 2; j < n; j++) {

count[i][j] = count[i][i] + count[i + 1][j];

s[i][j] = i;

for (int k = s[i][j - 1]; k <= s[i + 1][j]; k++) {

if (k < j) {

int ccount = count[i][k] + count[k + 1][j];

if (ccount < count[i][j]) {

s[i][j] = k;

count[i][j] = ccount;

}

}

}

count[i][j] += sum[j + 1] - sum[i];

}

}

System.out.println(count[0][n - 1]);

}

}