**普及组初赛模拟卷**

1. 单项选择题（共15题，每题2分，共计30分：每题有且仅有一个正确选项）
2. 在标准ASCII码表中，已知英文字母Z的ASCI 码十进制表示是 90，那么英文字母B的ASCI码二进制表示是（ ）。
3. 01000001 B. 01000010 C. 01000011 D. 01000000
4. 以下关于 CSP 与 NOIP 的描述正确的是（ ）。

A.CSP属于非专业认证，只有在校生才能参加

B.CSP是中国电子学会举办的程序设计竞赛

C.CSP 和 NOIP 毫无关系，没参加CSP也可以直接参加NOIP

D.CSP 和 NOIP 都是CCF 旗下的程序设计赛事

1. 以下不能用作 C++程序中的标识符的是（ ）。

A.private B. friends C.news D.pascal

1. NOI复赛测评机所用的 Linux系统属于(（ ）

A. UML B. IDE C. OS D. Database

1. 如果 65 536 种颜色用二进制编码来表示，至少需要（ ）个二进制位。。

A.16 B. 8 C.12 D.10

1. 搜索算法中的 BFS算法经常用到的数据结构是（ ）。
2. 堆 B. 栈 C.链表 D.队列
3. 在已经从小到大排好序的n元素单向链表中查询是否存在关键字为k的元素，最坏情况下运行的时间复杂度是（ ）。
4. O(logn) B. O(n) C. O(n2) D.O(nlogn)
5. 在下列各种排序算法中，不是以“比较”作为主要操作的算法是（ ）

A.归并排序 B.快速排序 C.冒泡排序 D. 桶排序

1. 关于计算机网络，下面的说法中正确的是（ ）。

A.现在的计算机必须连接到互联网才能正常运行

B.192.168.0.1是A类IP地址

C.互联网的诞生用到了现代计算机技术和现代通信技术

D. 接入互联网的计算机的IP地址已经全部升级到了IPv6 地址

1. 将(2,6,10,17)分别存储到某个地址区间为 0~10的哈希表中，如果哈希函数 h(x)=（ ），将不会产生冲突,其中 a%b 表示a除以b的余数，sqrt 表示开平方,floor表示向下取整。

A.x%11 B.x2%11 C.2x%11 D. floor(sqrt(x))%11

1. 现在有一个十六进制数 27，它等于二进制数的（ ）

A.100011 B.100101 C.100111 D. 100011

1. 以下逻辑表达式中，不管 A、B 如何取值，恒为假的是（ ）
2. (￢A∨B)∧(A∨B)∧A B.((￢A∨B)∨(A∨￢B))∧B

C. A∧((￢A∨B)∧(A∨￢B))∧￢A D. ((￢A∨B)∨(A∨￢B))∧A∧￢B

1. 某二叉树有16个结点都同时有左孩子结点和右孩子结点,则该二叉树中的叶子结点数是（ ）个。

A.19 B.17 C.18 D. 16

1. 现有 16张不同的卡片,其中红、黄、蓝、绿色卡片各4张。从中任取3张，要求红色最多有1张并且3张卡片不能是同一种颜色，不同的取法组合共有（ ）种。

A.232 B.472 C.256 D. 484

1. 有8个结点的非连通无向图最多有（ ）条边。

A.8 B.7 C.21 D. 49

1. 阅读程序（程序输入不超过数组成字符串定义的范围：判断题正确填√，错误填×；除特殊说明外，判断题1.5分，选择题3分，共计40分）

（1）

01 #include <bits/stdc++.h>

02

03 using namespace std;

04 pair<int,int> pi;

05

06 int parent[100000],sz[100000];

07

08 int n,m,components;

09

10 void initialize(){

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

12 parent[i] = i;

13 sz[i] = 1;

14 }

15 }

16

17 int find(int a){

18 if (parent[a] == a){

19 return a;

20 }

21 return find(parent[a]);

22 }

23

24 bool sameDSU(int a,int b){

25 return find(a) == find(b);

26 }

27

28 void merge(int a,int b){

29 a = find(a), b = find(b);

30 if (a == b) return ;

31 --components;

32 if (sz[a] > sz[b]) swap(a,b);

33 parent[a] = b;

34 sz[b] += sz[a];

35 }

36

37 int findsz(int a){

38 return sz[find(a)];

39 }

40

41 int main(){

42 cin >> n >> m;

43 initialize();

44 components = n;

45 int maxComponents = 1;

46 for (int i = 0; i < m; i++){

47 int a,b; cin >> a >> b;

48 merge(a,b);

49 maxComponents = max(maxComponents,findsz(a));

50 cout << components << " " << maxComponents <<endl;

51 }

52 }

判断题

1.如果输入的a和b从1开始编号，程序依然可以正常运行。（）

2.该程序计算了无向图连同分量数。（）

3.合并操作总是将较小树的根结点合并到较大树的根结点下。（）

4.为了确保该程序不会发生错误，需要保证输入的a和b互不相同。（）

选择题

5.在程序运行结束时，maxComponents变量可以存储的最大值是（）。

A.n/2 B.n C.m/2 D.m

6.如果合并操作merge总是将结点b合并到结点a，不论它们各自的sz大小如何，程序最坏运行时间会如何变化（）。

A.变快 B.保持不变 C.变慢 D.无法确定

（2）

01 #include <iostream>

02 using namespace std;

03 typedef long long i64;

04

05 const int NN= 1e7+5;

06 i64 ans，l，r;

07 bool S[NN];

08 int p[NN], O[NN],cnt;

09

10 inline void sss(){

11 o[1]= 1;

12 for(inti=2;i<= r; ++i){

13 if(!s[i])

14 o[p[++cnt]= i]= i;

15 for(int j=1;j<= cnt && p[j]\*i <= r; ++j){

16 s[p[j]\* i]= true;

17 o[p[j]\*i]=p[j];

18 if(i% p[j]== 0)break;

19 }

20 }

21}

22

23 inline void answer(){

24 sss();

25 for(inti=l ; i<= r; ++i)

26 if(s[i]) ans += i/o[i];

27 cout<< ans<< endl;

28 }

29

30 int main(){

31 cin >> l >> r;

32 answer();

33}

判断题

1.本题使用了埃氏筛法。 ( )

2.如果输人数据l比r大，那么程序会报错或者发生死循环。 ( )

3.本程序可以处理的最大r是10000000。 ( )

4.在确保l<r的前提下,l的大小不影响程序的运行速度。 ( )

选择题

5.当输人为"6 27”时，输出为( )

A. 115 B.116 C. 117 D. 118

6.本程序的时间复杂度为( )。

A. O(logN) B. O(N) C. O(loglogN) D. O(N loglogN)

（3）

01 #include <iostream>

02 #include <iomanip>

03 #include <math.h>

04 using namespace std;

05 pair<long long, long long> calc(int n, long long m){

06 if(n==0) return make\_pair(0,0);

07 long long len = 1ll <<(n-1),cnt = 1ll <<(2\*n-2);

08 pair<long long,long long> pos = calc(n-1, m % cnt);

09 long long x=pos.first,y= pos.second;

10 long long z=m/cnt;

11 if(z==0) return make\_pair(y，x);

12 if(z==1) return make\_pair(x, y+len);

13 if(z==2) return make\_pair(x+len, y+len);

14 if(z==3) return make\_pair(2\*len-y-1，len-x-1);

15 }

16 int main(){

17 int n, p, q;

18 cin >> n >> p >> q;

19 cout<< setprecision(3)

20 << sqrt(pow(1.0\*calc(n,p).first-calc(n,q).first,2.0)

21 + pow(1.0\*calc(n,p).second-calc(n,q).second,2.0))

22 << endl;

23 return 0;

24 }

判断题

1. 本程序编译会报warning ( )

2. 若想程序不出错，p、q不能超过22n。 ( )

3. 输入数据符合要求时，本程序的时间复杂度是0(max{N,logM})。 ( )

选择题

4.若输人为"2 0 10"，输出为( )。

A.4.243 B.3.00 C.4.24 D.3.606

5.当|p-q|=1时，不论输入的n是多少，只要p、q在符合要求的范围内，则输出为( )

A. 1 B.2 C.-1 D.0

6.对同一个n，共有几种输入，可以让输出达到最大?( )

A.2 B.4 C.1 D.8

三、完善程序

（1）（次大值之和）给定一个1到n的数字各出现一次的排列a[1]、a[2]、……、a[n]，定义f(l,r)表示a[l]、a[l+1]、a[l+2]、……、a[r]中的次大值，你需要求出对于所有的1<=i<=j<=n(n<=100000)，f(i,j)的和。

输入第一行包含一个整数n，第二n个整数表示a[i]，输出所有的f(i,j)之和。

如输入为3 1 2 3，则输出5;若输如为8 8 2 7 3 4 5 6 1，则输出136。

#include <bits/stdc++.h>

#define LL long long

using namespace std;

const int MAXN = 100005;

struct T {

int v, id; //v存放读入元素值，id存放读入顺序号

} x[MAXN];

int pre[MAXN], nxt[MAXN];

int cmp(T t1, T t2) { return t1.v < t2.v; }

void del(int p) {

int L = pre[p], r = nxt[p];

nxt[L] = r; pre[r] = L;

}

int main() {

int m, n, i, j, k;

scanf("%d", &n);

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

scanf("%d", &x[i].v);

x[i].id = ①;

pre[i] = i - 1;

nxt[i] = i + 1;

}

nxt[0] = 1;

pre[n + 1] = n;

②;

LL ans = 0;

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

LL L1, L2, r1, r2;

L1 = pre[x[i].id];

if (L1) L2 = pre[L1];

else L2 = -1;

r1 = nxt[x[i].id];

if (r1 != n + 1) r2 = nxt[r1];

else r2 = -1;

if (L2 != -1) ans += ③ \* i;

if (r2 != -1) ans += ④ \* i;

del(⑤);

}

cout << ans << endl;

}

1. ①处应填（ ）。

A. -1 B. 0 C. i D. n

2. ②处应填（ ）。

A. sort(x, x + n) B. sort(x, x + n, cmp)

C.sort(x, x + n + 1, cmp) D. sort(x + 1, x + n + 1, cmp)

3. ③处应填（ ）。

A. (L2 – L1) \* (L2 – x[i].id) B. (L1 – L2) \* (r1 – x[i].id)

C. (L1 – L2) \* (x[i].id – r1) D. (L2 – L1) \* (x[i].id – L1)

4. ④处应填（ ）。

A. (r2 – r1) \* (L1 – x[i].id) B. (r1 – r2) \* (r1 – x[i].id)

C. (r1 – r2) \* (x[i].id – r2) D. (r2 – r1) \* (x[i].id – L1)

5. ⑤处应填（ ）。

A. i B. L1 C. x[i].id D. r1

（2）（字典序最小最长路径）给出一张有n个结点、m条边的带权无环有向图，边以(ai,bi,li)的形式给出，表示从ai到bi的一条有向边，边权为li。现在要求对于每个结点，输出以它为起点且经过最多条边的路径经过的边数以及边权之和。如果有多条经过边数最多的路径，则选择将路径边权按顺序排列，得到序列中字典序最小的一条。

数据范围：

1≤n≤2×10^5,1≤m≤4×10^5,1≤ai,bi≤2×n,1≤li≤10^9

输入格式：

第1行是两个正整数n,m。第2行到第n+1行每行3个整数ai,bi,li,代表一条边。

01 #include <bits/stdc++.h>

02

03 using namespace std;

04

05 const int N = 2e5 + 10;

06 vector<pair<int, int>> g[N];

07 int dp[N], nexE[N], nexT[N], rnk[N], id[N];

08 long long sum[N];

09 bool vis [N];

10

11 void work(int cur) {

12 if (vis[cur]) return;

13 vis[cur] = true;

14 for (auto it : g[cur]) {

15 work(it.first);

16 dp[cur] = max(dp[cur], dp[it.first] + 1);

17 }

18 for (auto it : g[cur])

19 if(dp[it.first] + 1 == dp[cur] && (nexE[cur] == 0 || it.second < nexE[cur]))

20 ①;

21 }

22

23 signed main() {

24 ios::sync\_with\_stdio(false);

25 cin.tie(NULL);

26 int n, m;

27 cin >> n >> m; // 輸入节点数n和边数m

28 int u, v, w;

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

30 cin >>u >> v >> w; // 以u到V的一条有向边，边权为w

31 g[u].push\_back(make\_pair(v, w));

32 }

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

34 if( ② )

35 work(i);

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

37 id[i] = i;

38 sort(id + 1, id + n + 1, [&](int x, int y) { return ③； });

39 vector‹int> vec;

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

41 if(dp[id[i]] == 0) continue;

42 if(dp[id[i]] != dp[id[i - 1]] && vec.size()) {

43 sort(vec.begin(), vec.end(), [&](int x, int y) {

44 return nexE[x] < nexE[y] || ( ④ && rnk[nexT[x]] < rnk[nexT[y]]);

45 });

46 int cnt = 0;

47 for(int × : vec) rnk [x] = ++ cnt;

48 vec.clear();

49 }

50 vec.push\_back(id[i]);

51 for (auto it : g[id[i]])

52 if (dp[it.first] + 1 == dp[id[i]] && nexE[id[i]] == it. Second && (nexT[id[i]] == 0 || rnk[nexT[id[i]]] > rnk[it .first]))

53 nexT[id[i]] = it.first;

54 sum[id[i]] + = ⑤;

55 }

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

57 cout <<dp[i]<<' '<< sum[i] << endl;

58 return 0;

59 }

1. ①处应填（ ）

A. nexE[cur] = it.second B. nextE[cur] = dp[it.first] + 1

C. nexT[cur] = it.second D. nexT[cur] = it.first

2. ②处应填（ ）

A. dp[i] B. !vis[i] C. !rnk[i] D. nexT[i]

3. ③处应填（ ）

A. nexE[x] < nexE[y] B. nexT [x] < nexT[y]

C. dp[x] < dp[y] D. id[x] < id[y]

4. ④处应填（ ）

A. rnk[x] == rnk[y] B. rnk[x] < rnk[y]

C. nexE[x] > nexE[y] D. nexE[x] == nexE[y]

5. ⑤处应填（ ）

A. sum[id[i]] + nexE[id[i]]

B. sum[nexT[id[i]]] + nexE[id[i]]

C. sum[rnk[id[i]] + nexE[id[i]]

D. nexE[id[i]] + nexT[id[i]]