初赛模拟卷

1. 单项选择（共15题，每题2分，共计30分；每题有且仅有一个正确选项）
2. 下列不是linux命令的是（ ）

A. ps B. usd

C. cp D. cd

1. （ ）是目前互联网上常用的E-mail服务协议

A. POP3 B. FTP

C. HTTP D. Telnet

1. 在计算机中，防火墙的作用是（ ）

A. 防止火灾蔓延 B. 防止网络攻击

C. 防止计算机死机 D. 防止使用者误删数据

1. （ ）就是把一个复杂的问题分成两个或更多的相同类似的子问题，再把子问题分解成更小的子问题……直到最后的子问题可以简单地直接求解。而原问题的解就是子问题的并。

A. 贪心 B. 动态规划

C. 分治 D. 搜索

1. 如果平面上任取n个整点（横纵坐标均为整数），其中一定存在两个点，它们连线的中点是整点，则n至少是（ ）

A. 3 B. 4

C. 5 D. 8

1. 阅读下面的函数，当传入的三个参数为下列哪个时，函数能正常地获得返回值。（ ）

int dfs(int a, int b, int c) {

if(a == b) return 1;

int newa = b \* b / a, newb = c \* c / b, newc = a \* a / c;

return dfs(newa, newb, newc);

}

A. a = 0, b = 1, c = 2 B. a = 1, b = 2, c = 3

C. a = -1, b = 0, c = 1 D. a = 3, b = 4, c = 5

1. 由3个a,5个b和2个c构成的所有字符串中，包含子串为abc的共有（ ）个

A. 40320 B. 39600

C. 840 D. 780

1. 由4个不同的点构成的简单无向连通图的个数是（ ）

A. 32 B. 41

C. 35 D. 38

1. 一棵二叉树前序遍历为ABDECFGH，后序遍历为EDBGFHCA，以下不可能是其中序遍历的是（ ）

A. DEBAFGCH B. BEDAFGCH

C. EDBAGFCH D. DBEAFGCH

1. 数字319和数字377的最大公约数是（ ）

A. 33 B. 27

C. 29 D. 21

1. 前缀表达式- + \* 4 + 2 3 1 5的值为（ ）

A. 16 B. 19

C. 17 D. 15

1. 以下关于二叉树性质中，正确的描述的个数有（ ）

① 包含n个结点的二叉树的高度至少为ceil(log2n)

② 在任意一棵非空二叉树中，若叶子结点的个数为n0,出度为2的结点数为n2，则n0=n2+1

③ 深度为k的二叉树至多有2k个结点

④ 没有一棵二叉树的前序遍历序列与后序遍历序列相同

A. 0 B. 1

C. 2 D. 3

1. 将2个相同的红球，1个蓝球，1个白球放到10个编号不同的盒子中去，每个盒子最多放一个球，有多少种放法（ ）

A. 5040 B. 420

C. 2520 D. 1260

1. 设A和B是两个长度为n的有序数组，现在需要将A和B合并成一个排好序的数组，请问任何以元素比较作为基本运算的归并算法，在最坏情况下至少要做多少次比较？（ ）

A. n2 B. nlogn

C. 2n D. 2n - 1

1. G是一个非连通无向图（没有重边和自环），共有45条边，则该图至少有（ ）个顶点。

A. 8 B. 9

C. 10 D. 11

1. 阅读程序（程序输入不超过数组或字符串定义的范围；判断题选正误；除特殊说明外，判断题每题2分，选择题每题3分，共40分）
2. 阅读下列程序，回答问题。

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

02 using namespace std;

03 const int maxn = 110;

04 int n, k, w[maxn]; double v[maxn];

05 struct node {

06 double pt; int total;

07 bool operator < (const node &temp) const {

08 return pt < temp.pt;

09 }

10 };

11 priority\_queue<node> pq;

12 int main() {

13 cin >> n >> k; double ans = 0;

14 for(int i = 1;i <= n; ++i) cin >> v[i];

15 for(int i = 1;i <= n; ++i) cin >> w[i];

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

17 pq.push({v[i] / w[i], w[i]});

18 while(k > 0 && pq.size()) {

19 node u = pq.top(); pq.pop();

20 ans += min(k, u.total) \* u.pt;

21 k -= min(k, u.total);

22 }

23 cout << ceil(ans) << '\n';

24 return 0;

25 }

假设所有输入的数据均为[0, 100]之间的自然数，回答下列问题。

* 判断题

1. 存在可以被构造出的输入数据，使得程序无法正确运行。（ ）
2. 对于优先队列pq，当其非空时，若取出pq中的top处的元素，该元素必然是优先队列pq中pt属性较小的元素。（ ）
3. 当程序正常运行结束时，ans一定小于等于。（ ）

* 选择题

1. 当n = 5, k = 7, 数组v[i]为[1, 2, 3, 4, 5], w[i]为[5, 4, 3, 2, 1]时，程序的输出结果为（ ）

A. 13 B. 2

C. 12 D. 1

1. 阅读下列程序，回答问题。

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

02 using namespace std;

03 int main() {

04 int cnt[50]; memset(cnt, 0, sizeof cnt);

05 string str; cin >> str;

06 for(int i = 0;i < str.size(); ++i)

07 cnt[str[i] - 'a']++;

08 int ans = 0; bool judge = false;

09 for(int i = 0;i <= 25; ++i) {

10 if(cnt[i] % 2 == 0) ans += cnt[i];

11 else ans += cnt[i] - 1, judge = true;

12 }

13 if(judge) ++ans;

14 cout << ans << endl;

15 return 0;

16 }

假设输入的字符串仅包含小写字母，且长度不超过105，完成下列问题。

* 判断题

1. 将第09行的i <= 25改为i <= 30，不影响程序的结果。（ ）
2. 当第13行中的judge 为true时，求出的ans为偶数。（ ）

* 选择题

1. 当输入的字符串为helloworld时，输出的答案为（ ）

A. 7 B. 5

C. 8 D. 6

1. 阅读下列程序，回答问题。

**下面的代码的输入以下列方式给出：**

第一行两个正整数n, m,表示将给出一个长度为n的序列，并对其进行m次操作。其中，n和m的范围在[1, 1e6]之间。

第二行n个正整数，表示一个序列。

接下来m行，每行表示一种操作。

操作1给出的方式是 1 l r v，其中保证1 <= l <= r <= n，1 <= v <= 1e9，表示对[l, r]这个区间做某种和v有关的操作。

操作2给出的方式是 2 l r，其中保证1 <= l <= r <= n，表示对[l, r]这个区间做某种询问操作。

01 #include <cstdio>

02 #include <iostream>

03 #include <algorithm>

04 #include <cmath>

05 using namespace std;

06 const int maxn = 1e6 + 10;

07 long long tree[maxn \* 4], num[maxn];

08 void build(int u, int l, int r) {

09 if(l == r) {

10 tree[u] = num[l] - num[l - 1];

11 return;

12 }

13 int mid = (l + r) / 2;

14 build(u \* 2, l, mid);

15 build(u \* 2 + 1, mid + 1, r);

16 tree[u] = min(tree[u \* 2], tree[u \* 2 + 1]);

17 return;

18 }

19 void update(int u, int l, int r, int pos, long long v) {

20 if(l == r) {

21 tree[u] -= v;

22 return;

23 }

24 int mid = (l + r) / 2;

25 if(pos <= mid) update(u \* 2, l, mid, pos, v);

26 else update(u \* 2 + 1, mid + 1, r, pos, v);

27 tree[u] = min(tree[u \* 2], tree[u \* 2 + 1]);

28 return;

29 }

30 long long query(int u, int l, int r, int L, int R) {

31 if(L <= l && r <= R) return tree[u];

32 int mid = (l + r) / 2; long long t = 1e9;

33 if(L <= mid) t = min(t, query(u \* 2, l, mid, L, R));

34 if(R > mid) t = min(t, query(u \* 2 + 1, mid + 1, r, L, R));

35 return t;

36 }

37 int main() {

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

39 cin.tie(0); cout.tie(0);

40 int n, m; cin >> n >> m;

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

42 cin >> num[i];

43 build(1, 1, n);

44 for(int i = 1;i <= m; ++i) {

45 int opt; cin >> opt;

46 if(opt == 1) {

47 int l, r; long long v;

48 cin >> l >> r >> v;

49 update(1, 1, n, l, -v);

50 if(r + 1 <= n) update(1, 1, n, r + 1, v);

51 } else {

52 int l, r;

53 cin >> l >> r;

54 if(l == r) {

55 cout << "Yes\n";

56 } else {

57 long long ans = query(1, 1, n, l + 1, r);

58 cout << (ans >= 0 ? "Yes\n" : "No\n");

59 }

60 }

61 }

62 return 0;

63 }

* 判断题

1. 操作1是对区间进行减法操作，表示将区间[l, r]统一减去v。（ ）
2. 当进行操作2时，若区间长度为1，无论如何都会输出一行Yes。（ ）
3. 程序中32行的long long t = 1e9,改为long long t = 0仍然能正常求解。（ ）

* 选择题

1. (4分) 操作2判断的是对于区间[l, r]，当区间满足（ ）时，程序会输出Yes。

A. 单调增加 B. 单调减小

C. 单调不增 D. 单调不减

1. 阅读下列程序，回答问题。

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

02 using namespace std;

03 typedef long long ll;

04 ll l, r, f[25][15][15], a[25];

05 ll dfs(ll pos, bool limit, bool lead, ll pre1, ll pre2) {

06 if (!pos) {

07 return 1;

08 }

09 ll ans = 0;

10 if (!limit && !lead && f[pos][pre1 + 1][pre2 + 1] != -1) {

11 return f[pos][pre1 + 1][pre2 + 1];

12 }

13 ll up = limit ? a[pos] : 9;

14 for (ll i = 0; i <= up; ++i) {

15 if (i != pre1 && i != pre2) {

16 ans += dfs(pos - 1, limit && i == up, lead && !i, (!lead || i) ? i : -1, pre1);

17 }

18 }

19 if (!limit && !lead) {

20 f[pos][pre1 + 1][pre2 + 1] = ans;

21 }

22 return ans;

23 }

24 ll solve(ll x) {

25 ll cnt = 0;

26 while (x) {

27 a[++cnt] = x % 10;

28 x /= 10;

29 }

30 memset(f, -1, sizeof(f));

31 return dfs(cnt, 1, 1, -1, -1);

32 }

33 int main() {

34 cin >> l >> r;

35 cout << solve(r) - solve(l - 1);

36 return 0;

37 }

假设输入的l和r都是不超过1018的自然数，本程序目标是统计位于区间[l, r]中具有某种性质的正整数的数量。据此完成下列问题。

* 判断题

1. 将第07行改为return 0;程序统计出的数量不会改变。（ ）
2. (3分) 第15行的作用是保证对所有被统计到答案的数字中，不会出现连续的三个相等的正整数。（ ）
3. 当输入的l和r相等时，输出答案必定为1。（ ）

* 单选题

1. 若输入数据为1 100，则输出的答案为（ ）

A. 100 B. 90

C. 70 D. 80

1. (4分) 若输入数据为1 1000，则输出的答案为（ ）

A. 991 B. 688

C. 738 D. 832

1. 完善程序（单选题，每小题3分，共计30分）
2. 现在要求你求出满足下列要求的序列的数量，答案对109+7取模。

①

② ，满足 或

③ ，满足

输入数据格式为第一行一个正整数t，表示有t组询问。

对每组询问给出一行四个正整数n, p, x, y，以空格分开。

保证。对每组询问你都需要回答一行正整数表示答案。

#include <cstdio>

#include <iostream>

#include <algorithm>

#include <cmath>

#include <cstring>

using namespace std;

const int maxn = 1e4 + 10;

const int mod = 1e9 + 7;

int n, dp[2 \* maxn];

long long p, x, y;

int main( ) {

int t; cin >> t;

while(t--) {

int ans = 0;

cin >> n >> p >> x >> y;

if(x == y) {

bool judge = true;

for(int i = 1;i <= n; ++i) if(\_\_\_\_\_①\_\_\_\_\_) judge = false;

cout << \_\_\_\_\_②\_\_\_\_\_ << '\n';

continue;

}

dp[maxn] = 1;

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

for(int j = 0;j <= i; ++j) {

int k = \_\_\_\_\_③\_\_\_\_\_; dp[maxn + k] = 0;

if(\_\_\_\_\_④\_\_\_\_\_)

dp[maxn + k] = (\_\_\_\_\_⑤\_\_\_\_\_) % mod;

if(i == n) ans = (ans + dp[maxn + k]) % mod;

}

}

cout << ans << '\n';

for(int i = -n; i <= n; ++i) dp[maxn + i] = 0;

}

return 0;

}

1. ①处应填（ ）

A. x % p == 0 B. x \* i % p == 0

C. i % p == 0 D. judge && (x % p == 0)

1. ②处应填（ ）

A. judge ? 1 : 0 B. judge ? n : 0

C. judge ? 0 : 1 D. judge ? 0 : n

1. ③处应填（ ）

A. j B. j + (i – j)

C. i - j D. j – (i – j)

1. ④处应填（ ）

A. (x \* i + y \* j) % p == 0

B. (x \* i + y \* j) % p != 0

C. (x \* j + y \* (i – j)) % p == 0

D. (x \* j + y \* (i – j)) % p != 0

1. ⑤处应填（ ）

A. dp[maxn + k + 1] + dp[maxn + k - 1]

B. dp[maxn + k + 1] - dp[maxn + k - 1]

C. dp[k + 1] + dp[k - 1]

D. dp[k + 1] - dp[k - 1]

1. 现在你有一个包含n个不同的字符串的串集S。现在询问你有多少个不同的长度为m的字符串，使得字符串中至少有一个连续子串，位于串集S中。由于答案可能很大，你需要对10007取模。

所有出现的字符均只考虑大写字母。

数据范围有，每个字符串的串长

输入格式为第一行两个正整数n,m，表示你拥有的串集大小为n，你可以构造的字符串的长度为m。

接下来n行，每行一个字符串，表示串集中的一个字符串。

**提示：考虑用所有字符串的数量减去非法字符串的数量。**

#include<bits/stdc++.h>

using namespace std;

const int maxn = 6000 + 10;

const int mod = 10007;

struct Aho\_corasick{

int ch[maxn][30], f[maxn], tot;

bool endp[maxn];

Aho\_corasick() {

memset(ch, 0, sizeof ch);

memset(endp, 0, sizeof endp);

memset(f, 0, sizeof f);

tot = 0;

}

void insert(string &s) {

int n = s.size(); int u = 0;

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

if(!ch[u][s[i] - 'A'])

ch[u][s[i] - 'A'] = ++tot;

u = ch[u][s[i] - 'A'];

}

\_\_\_\_\_①\_\_\_\_\_;

}

void getfail() {

queue<int> q;

for(int i = 0;i <= 25; ++i)

if(ch[0][i])

q.push(ch[0][i]);

while(!q.empty()) {

int u = q.front(); q.pop();

for(int i = 0;i <= 25; ++i) {

int v = ch[u][i];

if(v) {

f[v] = ch[f[u]][i];

\_\_\_\_\_②\_\_\_\_\_

q.push(v);

} else {

ch[u][i] = ch[f[u]][i];

}

}

}

}

} AC;

int quickpow(int x, int y) {

int base = x, ans = 1;

while(y) {

if(y & 1) \_\_\_\_\_③\_\_\_\_\_

base \*= base;

ans %= mod; base %= mod;

y /= 2;

}

return ans;

}

int ans, dp[105][maxn];

int main() {

int n, m; cin >> n >> m;

ans = quickpow(26, m);

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

string str; cin >> str;

AC.insert(str);

}

AC.getfail();

int max\_state = AC.tot;

dp[0][0] = 1;

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

for(int state = 0;state <= max\_state; ++state) {

for(int nxt = 0;nxt <= 25; ++nxt) {

if(\_\_\_\_\_④\_\_\_\_\_) continue;

\_\_\_\_\_⑤\_\_\_\_\_ += dp[i][state];

\_\_\_\_\_⑤（两处⑤所填内容相同）\_\_\_\_\_ %= mod;

}

}

}

for(int state = 0; state <= max\_state; ++state)

ans = (ans + mod - dp[m][state]) % mod;

cout << ans << '\n';

return 0;

}

1. ①处应填（ ）

A. endp[u] = 0 B. endp[u] = 1

C. f[u] = 0 D. f[u] = 1

1. ②处应填（ ）

A. if(endp[f[u]]) endp[u] = 1;

B. if(endp[f[u]]) endp[v] = 1;

C. if(endp[f[v]]) endp[u] = 1;

D. if(endp[f[v]]) endp[v] = 1;

1. ③处应填（ ）

A. ans \*= base; B. base \*= ans

C. ans += base D. base += ans

1. ④处应填（ ）

A. AC.f[AC.ch[state][nxt]]

B. AC.endp[AC.ch[state][nxt]]

C. AC.f[AC.ch[nxt][state]]

D. AC.endp[AC.ch[nxt][state]]

1. ⑤处应填（ ）

A. dp[i][AC.f[AC.ch[state][nxt]]]

B. dp[i + 1][AC.f[AC.ch[state][nxt]]]

C. dp[i][AC.ch[state][nxt]]

D. dp[i + 1][AC.ch[state][nxt]]