2016级数据结构第九次上机解题报告

A

直接对原序列进行排序，输出相应位置的元素即可。

参考代码：

#include <iostream>

#include <string>

#include <cstring>

#include <algorithm>

using namespace std;

const int MaxN = (int) 1e6;

const int MaxNum = 10000;

int n, M;

int Tel[MaxN];

int main() {

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

cin.tie(0);

cout.tie(0);

cin >> n >> M;

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

cin >> Tel[i];

}

sort(Tel, Tel + n);

cout << Tel[M - 1] << "\n";

return 0;

}

B

每个数都可以被分解成最小素因数\*其他因数

根据这个性质，使用一个小根堆，每次找到第i个丑数Number[i]

就把Number[i]\*Prime[]这K个数放到堆里

参考代码：

#include <iostream>

#include <ctime>

#include <cstdlib>

#include <string>

#include <cstring>

#include <algorithm>

#include <climits>

#include <map>

#include <vector>

#include <queue>

using std::cin;

using std::cout;

using std::map;

using std::priority\_queue;

const int MaxK = 20;

const int MaxN = 100000;

const int MaxNum = 100;

int K, N, now;

int nums[MaxK];

long long prev;

priority\_queue<long long, std::vector<long long>, std::greater<long long> > pq;

map<long long, bool> m;

int main() {

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

cin.tie(0);

cout.tie(0);

cin >> K >> N;

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

cin >> nums[i];

pq.push((long long) nums[i]);

}

while (--N) {

prev = pq.top();

pq.pop();

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

if (INT\_MAX / nums[i] < prev) break;

pq.push(prev \* nums[i]);

if (prev % nums[i] == 0) break;

}

}

cout << pq.top() << "\n";

return 0;

}

C

对于每一个任务，算出其与给定区间的交集的长度，乘以单位时间内的工作量即可。

参考代码：

#include <iostream>

#include <string>

#include <cstring>

#include <algorithm>

using namespace std;

const int MaxN = (int) 6e5;

const int MaxNum = 10000;

int n, T1, T2, B, E, W, tot;

int Tel[MaxN];

#define Min(a, b) (((a) < (b)) ? (a) : (b))

#define Max(a, b) (((a) > (b)) ? (a) : (b))

int main() {

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

cin.tie(0);

cout.tie(0);

tot = 0;

cin >> n >> T1 >> T2;

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

cin >> W >> B >> E;

if (B <= T2 && E >= T1) {

tot += (Min(E, T2) - Max(B, T1) + 1) \* W;

}

}

cout << tot;

return 0;

}

D

这道题题意是求根节点到最近子节点的距离，看起来和求树的深度差不多，不过我们要注意一个特殊地方，也就是当只有一个子树为空的时候，应该返回另一个子树的深度加一，而不是返回1。

参考代码：

#include <iostream>

#include <cstdio>

using namespace std;

struct Tree {

int val;

Tree \*left, \*right;

Tree() : val(0), left(nullptr), right(nullptr) {}

Tree(int x) : val(x), left(nullptr), right(nullptr) {}

};

void preOrder(Tree \*tree) {

if (tree) {

printf("%c", tree->val);

preOrder(tree->left);

preOrder(tree->right);

}

}

void deleteTree(Tree \*tree) {

if (tree) {

deleteTree(tree->left);

deleteTree(tree->right);

delete tree;

}

}

int countLeaf(Tree \*tree) {

if (!tree) return 0;

if (!tree->left and !tree->right) return 1;

return countLeaf(tree->left) + countLeaf(tree->right);

}

char a[1010];

void createTree(Tree \*&t, int& i) {

char p = a[i++];

if (p == '\0') {

return;

}

if (p != '#') {

t = new Tree(p);

createTree(t->left, i);

createTree(t->right, i);

}

}

int minHeight(Tree\* root) {

if (!root) return 0;

if (!root->left) return 1 + minHeight(root->right);

if (!root->right) return 1 + minHeight(root->left);

return 1 + min(minHeight(root->left), minHeight(root->right));

}

int main() {

while (~scanf("%s", a)) {

int i = 0;

Tree \*root = nullptr;

createTree(root, i);

printf("%d\n", minHeight(root));

deleteTree(root);

}

return 0;

}

E

dp，先不考虑第二个限制条件，那么数量就是 ，再考虑第二个限制条件，我们发现正着考虑比较复杂，就从反面考虑，找到边全都小于d的路径数量就好了，那这个就是dp[n][d−1]，所以两个数减一下就好了，注意取模

参考代码：

#include <cstdio>

#define mod 1000000007

int dp[101][101];

int main()

{

//freopen("in.in","r",stdin);

//freopen("out.out","w",stdout);

int n,k,d;

for(int i=1;i<=100;i++)

dp[0][i]=1;

for(int i=1;i<=100;i++)

for(int j=1;j<=100;j++)

for(int k=1;k<=j;k++)

{

if(i-k>=0)

dp[i][j]=(dp[i][j]+dp[i-k][j])%mod;

else break;

}

while(scanf("%d%d%d",&n,&k,&d)!=EOF)

printf("%d\n",(dp[n][k]-dp[n][d-1]+mod)%mod);

return 0;

}

F

假设一组数据中数字分别是a1, a2, ..., an, m，每个距离移动的次数分别是x1, x2, ..., xn，移动m个单位长度的次数是xn+1，那么要满足已知条件只需满足方程：

a1x1+a2x2+...+anxn+mxn+1 = 1 有解，即：

gcd (a1, a2, ..., an, m) = 1，接下来对m进行素因子分解，然后排除公因子非1的情况即可。

设g为公因子非1的情况数，f(i)表示有i个公因子的情况数，由容斥原理得：g = f(1) - f(2) + f(3) -... f(k)

参考代码：

#include <iostream>

#include <cstring>

using namespace std;

#define LL unsigned long long

int fac[35], k, a[20], n, m, x;

LL tp;

LL my\_pow (LL a, int b) //计算a^b

{

LL res = 1;

while (b--) res \*= a;

return res;

}

void dfs (int pos, int cnt, int num) //dfs

{

if (cnt == num)

{

x = m;

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

x /= a[i]; //x/p表示[1,x]中有多少个数是p的倍数

tp += my\_pow (x, n); //要选n个数，每个数有x种选择

return ;

}

for (int i = pos; i < k; i++)

{

a[cnt] = fac[i];

dfs (i+1, cnt+1, num);

}

}

void divide (int p) //分解素因子，fac存放p的所有素因子

{

for (int i = 2; i \* i <= p; i++)

{

if (p % i == 0)

{

fac[k++] = i;

p /= i;

while (p % i == 0)

p /= i;

}

}

if (p > 1) fac[k++] = p;

}

int main()

{

LL ans, g;

int i;

while (cin >> n >> m)

{

g = 0;

memset(fac, 0, sizeof(fac));

memset(a, 0, sizeof(a));

k = 0;

x = 0;

divide (m);

for (i = 1; i <= k; i++) //g = f(1)-f(2)+f(3)-f(4)+...f(k)

{

tp = 0;

dfs (0, 0, i);

if (i & 1) g += tp;

else g -= tp;

}

ans = my\_pow (m, n) - g; //ans = m^n - g

cout << ans << endl;

}

return 0;

}

G

用map将人名映射成数字，然后进行dfs代码。

参考代码：

#include <cstdio>

#include <iostream>

#include <string>

#include <map>

#include <vector>

using namespace std;

map<string,int> hashh;

int mp[205][205];

int n,ans;

string change(string s)

{

for(int i=0;i<s.length();i++)

if(s[i]>='A'&&s[i]<='Z')

s[i]-='A'-'a';

return s;

}

void dfs(int x,int from,int step)

{

if(step>ans)

ans=step;

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

if(mp[x][i]==1&&i!=from)

dfs(i,x,step+1);

return;

}

int main()

{

string s1,s2;

while(cin>>n)

{

cin>>s1;

cin>>s2;

cin>>s2;

hashh[change(s1)]=1;

hashh[change(s2)]=2;

mp[1][2]=1;

mp[2][1]=1;

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

{

cin>>s1;

cin>>s2;

cin>>s2;

hashh[change(s1)]=i+2;

mp[hashh[change(s2)]][i+2]=1;

mp[i+2][hashh[change(s2)]]=1;

}

dfs(2,0,1);

cout<<ans<<endl;

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

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

mp[i][j]=0;

hashh.clear();

ans=0;

}

return 0;

}