

Data structure

reference



2018-12-8

Firerabbit

FZU

目录

[DEV-CPP 设置及快捷键 2](#_Toc532288972)

[quick read() 2](#_Toc532288973)

[素数筛法 3](#_Toc532288974)

[射击游戏 3](#_Toc532288975)

[好的序列 4](#_Toc532288976)

[mountain① (810, 13044) ===== stack bsearch 5](#_Toc532288977)

[mountain② (60, 2204) quick read 6](#_Toc532288978)

[谁是老大 7](#_Toc532288979)

[字符串排序 （桶排序+桶内排序） 8](#_Toc532288980)

[人潮最多的时段 11](#_Toc532288981)

[spruce 13](#_Toc532288982)

[message ① （pointer） 14](#_Toc532288983)

[message ② （vector） 16](#_Toc532288984)

[前缀游戏 （字典树）① 18](#_Toc532288985)

[前缀游戏2 （计算好需要最大空间） 21](#_Toc532288986)

[前缀游戏3（二叉搜素树） 23](#_Toc532288987)

[人口普查 （hash） 27](#_Toc532288988)

[集合1 29](#_Toc532288989)

[集合2 ① 30](#_Toc532288990)

[集合2 ② 31](#_Toc532288991)

[摸鱼 32](#_Toc532288992)

[智慧果分堆 35](#_Toc532288993)

[朋友圈 （并查集） 36](#_Toc532288994)

[破碎的数组 (并查集 + offline) 38](#_Toc532288995)

[最短路径 (forward star) 40](#_Toc532288996)

[最短路径2 （邻接表） 42](#_Toc532288997)

[贪心大法官 （拓扑排序） 44](#_Toc532288998)

[二叉树遍历 46](#_Toc532288999)

[找某个字符串的不同子串的数目(O(n^2)) 48](#_Toc532289000)

[二分查找 48](#_Toc532289001)

[最小生成树（KRUSKAL） 50](#_Toc532289002)

DS

# DEV-CPP 设置及快捷键

工具-> 编辑器选项 –> 高亮当前行

* 语法 – > Symbol（前景 Teal）, Space, String 背景：（217， 217， 217）
* Selected text (2, 6) 第二行，第六列

工具-> 编译选项 -> 代码生成/优化 –> 代码生成 -> 语言标准 -> GNU C++11

Ctrl + D delete one line

Ctrl + E copy one line to below

Ctrl + I Increase Search

Ctrl + R replace

Ctrl + T To-Do /\* TODO (xjliang#1#): test for to do \*/

# quick read()

// quick read positive integers

inline int read()

{

int data = 0;

char ch = 0;

while (ch<'0' || ch>'9') ch = getchar(); // read other char

while (ch>='0' && ch<='9') data = data\*10 + ch-'0', ch=getchar();

return data;

}

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

cin.tie(NULL);

# 素数筛法

#include <stdio.h>

#define N 100005

short is\_not\_prime[N] = { 0 }; // all is prime

void Sieve()

{

int m, k;

for (m = 2; m \* m <= N; m++)

{

if (!is\_not\_prime[m])

{

for (k = m + m; k <= N; k += m)

{

if (!is\_not\_prime[k])

{

is\_not\_prime[k] = 1;

}

}

}

}

}

# 射击游戏

#include <stdio.h>

#define MAX\_N 3003

typedef struct Pair

{

int id;

int key;

} next[MAX\_N];

void JosePhus(int n)

{

int i, j, k, m = 1;

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

next[i].id = i + 1;

k = n - 1;

next[k].id = 0; // 组成循环数组

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

{

for (j = 1; j < m; j++)

k = next[k].id;

m = next[next[k].id].key;

printf("%d ", next[k].id + 1);

next[k].id = next[next[k].id].id;

}

printf("%d", next[k].id + 1);

}

# 好的序列

#include <stdio.h>

#include <string.h>

#define N 100005

char str[N];

int main(void)

{

int i, j, len;

int odd\_a = 0, even\_a = 0;

int odd\_b = 0, even\_b = 0;

int res\_odd = 0, res\_even = 0;

scanf("%s", str);

len = strlen(str);

for (i = 0; i < len; i += 2)

{

if (str[i] == 'a') odd\_a++;

else odd\_b++;

}

for (i = 1; i < len; i += 2)

{

if (str[i] == 'a') even\_a++;

else even\_b++;

}

res\_odd = (odd\_a - 1) \* odd\_a / 2 + (even\_a - 1) \* even\_a / 2

+ (odd\_b - 1) \* odd\_b / 2 + (even\_b - 1) \* even\_b / 2

+ len;

res\_even = odd\_a \* even\_a + odd\_b \* even\_b;

printf("%d %d", res\_even, res\_odd);

return 0;

}

# mountain① (810, 13044) ===== stack bsearch

#include <iostream>

#include <cstdio>

const int maxn = 5000000 + 5;

typedef long long ll;

int s[maxn];

int top = -1;

int main()

{

int n, v;

scanf("%d", &n);

ll sum = 0;

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

{

scanf("%d", &v);

if (top >= 0 && s[top] <= v)

{

int lo = 0, hi = top, m;

while (lo <= hi) // bsearch

{

m = (lo + hi) >> 1;

if (s[m] <= v) hi = m - 1;

else lo = m + 1;

}

top = hi;

}

s[++top] = v;

sum += top;

}

printf("%lld", sum);

return 0;

}

# mountain② (60, 2204) quick read

#include <stdio.h>

#define N 5000005

int stack[N];

int main()

{

long long sum = 0;

int hi;

int n;

int size = 0;

scanf("%d", &n);

while (n-- > 0)

{

scanf("%d", &hi);

while (size > 0 && hi >= stack[size - 1])

sum += --size;

stack[size++] = hi;

}

sum += (size - 1) \* size / 2;

printf("%lld", sum);

return 0;

}

# 谁是老大

#include <stdio.h>

#define N 1000

#define M 3

int next[N];

void Josephus(int n, int m)

{

int i, j, k;

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

next[i] = i + 1;

k = n - 1;

next[k] = 0;

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

{

for (j = 1; j < m; j++)

k = next[k];

next[k] = next[next[k]];

}

printf("%d", next[k] + 1);

}

int main(void)

{

int n;

scanf("%d", &n);

Josephus(n, M);

return 0;

}

# 字符串排序 （桶排序+桶内排序）

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define N 100

typedef struct Info

{

char str[N + 1];

struct Info \*next;

} Node;

typedef Node \*Link;

Link bottom[N + 1];

Link top[N + 1];

int count[N + 1] = { 0 };

// swap int

#define swap(a, b) do { int t = a; a = b; b = t; } while (0)

// swap pointer Node\*

#define swap2(p1, p2) do { Link p = p1; p1 = p2; p2 = p; } while (0)

Node \*CreatNode()

{

Node \*p = (Node \*)malloc(sizeof(Node));

p->next = NULL;

return p;

}

void BinSort(int n)

{

int i, j, b;

Link p = NULL;

memset(bottom, 0, sizeof(bottom));

memset(top, 0, sizeof(top));

// 创建桶

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

{

p = CreatNode();

scanf("%s", p->str);

b = strlen(p->str);

if (bottom[b]) // Bin is not empty

{

top[b]->next = p;

top[b] = p;

} else {

bottom[b] = top[b] = p;

}

count[b]++;

}

for (i = 1; i < N; i++)

{

for (j = i + 1; j <= N; j++)

{

if (count[i] < count[j] ||

(count[i] == count[j] && count[i] && strlen(bottom[i]->str) > strlen(bottom[j]->str)))

{

swap(count[i], count[j]);

swap2(bottom[i], bottom[j]);

}

}

}

// Output

for (b = 1; b <= N; b++)

{

if (bottom[b])

{

printf("%d", count[b]);

for (p = bottom[b]; p; p = p->next)

printf(" %s", p->str);

putchar('\n');

}

}

}

int main()

{

int n;

scanf("%d", &n);

BinSort(n);

return 0;

}

# 人潮最多的时段

#include <cstdio>

#include <algorithm>

const int N = 100000 + 5;

int in[N], out[N];

int main()

{

int n;

scanf("%d", &n);

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

scanf("%d %d", &in[j], &out[j]);

}

std::sort(in, in + n);

std::sort(out, out + n);

int lastDelta = 0, lastStart = -1;;

int inI = 0, outI = 0;

int ans = 0, t1 = 1, t2 = 1;

int delta = 0;

for (int t = in[0]; t <= out[n-1]; t++)

{

for(; inI < n && in[inI] <= t; inI++) { }

for(; outI < n && out[outI] <= t; outI++) { }

delta = inI - outI;

if (delta != lastDelta)

{

if (ans < lastDelta)

{

ans = lastDelta;

t1 = lastStart;

t2 = t;

}

lastStart = t;

lastDelta = delta;

}

}

printf("%d-%d %d", t1, t2, ans);

return 0;

}

# spruce

#include <cstdio>

#include <cstring>

using namespace std;

const int N = 1000 + 5;

const int M = 3;

int fa[N], son\_cnt[N];

int main()

{

int n;

scanf("%d", &n);

fa[1] = 0;

int maxFa = 1;

// memset(son\_cnt, 0, sizeof(son\_cnt)); global varaible

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

scanf("%d", &fa[i]);

son\_cnt[fa[i]]++;

if (maxFa < fa[i])

maxFa = fa[i];

}

bool ans = true;

for (int i = maxFa; i > 0; --i) {

if (son\_cnt[i] == 0) continue;

if (son\_cnt[i] > 0 && son\_cnt[i] < M) {

printf("No");

ans = false;

break;

}

son\_cnt[fa[i]]--;

}

if (ans) printf("Yes");

return 0;

}

# message ① （pointer）

#include <stdio.h>

#include <stdlib.h>

#define N 200005

// treeNode

typedef struct node

{

int num;

struct node\* next;

}no, \*np;

np head[N];

np tail[N];

// stack

int top;

int cot; // counter

struct

{

int cur;

np curp;

}stack[N];

int index[N];

int node[N];

int end[N];

int main()

{

int n, q, t;

scanf("%d %d", &n, &q);

np ptr;

int i;

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

head[i] = NULL;

for (i = 2; i <= n; ++i)

{

scanf("%d", &t);

// insert

ptr = (np)malloc(sizeof(no));

ptr->num = i;

ptr->next = NULL;

if (head[t])

{

tail[t]->next = ptr;

tail[t] = tail[t]->next;

} else {

head[t] = tail[t] = ptr;

}

}

top = 1;

cot = 1;

index[1] = 1;

node[1] = 1;

stack[top].cur = 1;

stack[top].curp = head[1];

while (top > 0)

{

ptr = stack[top].curp;

if (stack[top].curp)

{

stack[top].curp = ptr->next;

top++;

stack[top].cur = ptr->num;

stack[top].curp = head[stack[top].cur];

index[ptr->num] = ++cot;

node[cot] = ptr->num;

} else {

end[stack[top].cur] = cot;

top--;

}

}

int u, k;

while (q--)

{

scanf("%d %d", &u, &k);

if (end[u] - index[u] < k - 1)

{

printf("-1\n");

} else {

printf("%d\n", node[index[u] + k - 1]);

}

}

return 0;

}

# message ② （vector）

#include <cstdio>

#include <vector>

using std::vector;

const int N = 200005;

vector<int> tree[N];

int index[N];

int node[N];

int end[N];

int cot; // counter

void Preorder(int cur)

{

cot++;

index[cur] = cot;

node[cot] = cur;

for (int son = 0; son < tree[cur].size(); son++) // traverse the son node

{

Preorder(tree[cur][son]);

}

end[cur] = cot;

}

int main()

{

int n, q, t, i;

int u, k;

scanf("%d %d", &n, &q);

for (i = 2; i <= n; ++i)

{

scanf("%d", &t);

tree[t].push\_back(i);

}

cot = 0;

node[1] = 1;

index[1] = 1;

Preorder(1);

while (q--)

{

scanf("%d %d", &u, &k);

if (k > end[u] - index[u] + 1) {

printf("-1\n");

} else {

printf("%d\n", node[index[u] + k - 1]);

}

}

return 0;

}

# 前缀游戏 （字典树）①

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#define N 500005

typedef struct Node

{

int cnt;

char ch;

struct Node \*head;

struct Node \*next;

} no, \*np;

np FindChar(const np sons, int ch)

{

np p = sons->head;

while (p && p->ch != ch)

p = p->next;

return p;

}

np NewNode(char ch)

{

np p = (np)malloc(sizeof(no));

p->cnt = 0;

p->ch = ch;

p->head = p->next = 0;

return p;

}

void Insert(np root, const char\* word)

{

np p = root;

np q;

while (\*word)

{

if ((q = FindChar(p, \*word)) == NULL) {

q = NewNode(\*word);

q->next = p->head;

p->head = q;

}

p = q;

p->cnt += 1;

++word;

}

}

int Search(const np root, const char\* word)

{

np p = root;

while (\*word && p) {

p = FindChar(p, \*word);

++word;

}

return (!p ? 0 : p->cnt);

}

int main()

{

int n, i;

np root = NewNode('\0');

scanf("%d", &n);

char word[31];

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

scanf("%s", word);

Insert(root, word);

}

scanf("%d", &n);

while (n--)

{

scanf("%s", word);

printf("%d\n", Search(root, word));

}

// free root

return 0;

}

# 前缀游戏2 （计算好需要最大空间）

#include <stdio.h>

typedef struct node

{

int ctr;

char ch;

int son;

int nex;

} no;

#define N 50000+(30-4)\*50000+26\*2 // ??

#define M 26

#define D 97 //'a'

no mal[N];

int mp = 0;

int main(void)

{

char s[39];

int i, j, p, f, n, m;

for( i=1; i<=M; i++ )

{

mal[i].ctr = 0;

mal[i].ch = D + i - 1;

mal[i].son = 0;

mal[i].nex = 0;

}

mp = M+1;

scanf("%d", &n);

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

{

scanf("%s", s);

f = s[0] - D + 1;

mal[f].ctr++;

p = mal[f].son;

j = 1;

while (s[j])

{

while(p && mal[p].ch != s[j])

p = mal[p].nex;

if (p) // find some common prefix

{

mal[p].ctr++;

f = p;

p = mal[p].son;

}

else // can't find

{

mal[mp].ctr = 1;

mal[mp].ch = s[j];

mal[mp].son = 0;

mal[mp].nex = mal[f].son; // front insert to the son list

mal[f].son = mp;

f = mp;

mp++; // memory allocated

p = 0; // no common prefix, will always insert new node

}

j++;

}

}

scanf("%d", &m);

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

{

scanf("%s", s);

f = s[0] - D + 1;

j = 1;

p = mal[f].son;

while(p && s[j])

{

while(p && mal[p].ch != s[j]) // search in the brother list

p = mal[p].nex;

if(p)

{

f = p;

p = mal[p].son;

j++;

}

}

printf("%d\n", !s[j] ? mal[f].ctr : 0); // !s[j] => find the prefix

}

return 0;

}

# 前缀游戏3（二叉搜素树）

// 260ms, 6524kb

#include <iostream>

#include <string>

#include <cstring>

using namespace std;

struct Prefix

{

string S;

Prefix \*left = NULL;

Prefix \*right = NULL;

};

void Judge(const Prefix \*p, const string& a);

int c; // global counter used for recusive function

int main()

{

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

cin.tie(NULL);

int n, i;

string a;

Prefix \*root, \*p, \*q;

root = new Prefix;

root->S = "\0";

p = root;

cin >> n;

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

{

cin >> a;

q = new Prefix;

q->S = a;

p = root;

while (true)

{

if (strcmp(a.c\_str(), p->S.c\_str()) < 0)

{

if (p->left)

{

p = p->left;

}

else

{

p->left = q;

break;

}

}

else // >=

{

if (p->right)

{

p = p->right;

}

else

{

p->right = q;

break;

}

}

}

}

cin >> n;

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

{

cin >> a;

c = 0;

p = root->right;

Judge(p, a);

cout << c << endl;

}

return 0;

}

void Judge(const Prefix \*p, const string& a)

{

if (strncmp(a.c\_str(), p->S.c\_str(), a.size()) == 0)

{

c++;

}

if (strncmp(a.c\_str(), p->S.c\_str(), a.size()) >= 0 && p->right)

{

Judge(p->right, a);

}

else if (strncmp(a.c\_str(), p->S.c\_str(), a.size()) <= 0 && p->left)

{

Judge(p->left, a);

}

}

# 人口普查 （hash）

#include <iostream>

#include <map>

using namespace std;

typedef struct Pair

{

int w, h;

Pair(int a, int b):w(a), h(b) { }

friend bool operator<(const Pair& lhs, const Pair& rhs)

{

return (lhs.w < rhs.w

|| (lhs.w == rhs.w && lhs.h < rhs.h));

}

} Pair;

inline int Gcd(int a, int b)

{

if (a % b == 0) return b;

return Gcd(b, a % b);

}

void Init(int n, map<Pair, int>& mp)

{

int a, b , c, d;

double k;

int w, h;

int gcd;

int squre\_cnt = 0;

while (n-- > 0)

{

cin >> a >> b >> c >> d;

if (c - a == d -b)

{

squre\_cnt++;

continue;

}

w = c - a;

h = d - b;

gcd = Gcd(w, h);

w /= gcd;

h /= gcd;

if (mp.find(Pair(w, h)) == mp.end())

mp.insert(make\_pair(Pair(w, h), 1));

else

mp[Pair(w, h)]++;

}

cout << "1:1 have " << squre\_cnt << endl;

}

int main()

{

map<Pair, int> mp;

int n;

cin >> n;

Init(n, mp);

map<Pair, int>::iterator endIter = mp.end();

cout << mp.size() << endl;

for (map<Pair, int>::iterator iter = mp.begin(); iter != endIter; iter++)

{

cout << iter->first.w << ":" << iter->first.h << " have " << iter->second << endl;

}

return 0;

}

# 集合1

#include <cstdio>

#include <set>

using namespace std;

int main(void)

{

int n;

scanf("%d", &n);

int x;

long long sum = 0;

set<int> set;

while (n-- > 0)

{

scanf("%d", &x);

if (set.find(x) == set.end())

{

set.insert(x);

sum += x;

}

else

{

set.erase(x);

sum -= x;

}

printf("%lld", sum);

}

return 0;

}

# 集合2 ①

#include <cstdio>

#include <set>

using namespace std;

int main(void)

{

int n, x;

char ch;

multiset<int> set;

multiset<int>::iterator iter;

scanf("%d", &n);

while (n-- > 0)

{

scanf("%d %c", &x, &ch);

if (ch == '+')

{

iter = set.lower\_bound(x + 1);

set.erase(iter, set.end());

set.insert(x);

}

else // ch == '-'

{

iter = set.upper\_bound(x - 1);

set.erase(set.begin(), iter);

set.insert(x);

}

}

long long ans = 0;

for (multiset<int>::iterator it = set.begin(); it != set.end(); ++it)

ans += \*it;

printf("%lld", ans);

return 0;

}

# 集合2 ②

#include <stdio.h>

#include <string.h>

#define N 500001

#define M 1000000001

typedef long long ll;

ll num[N];

char sign[N];

int main(void)

{

int n, f = 0, r = M, i;

ll ans = 0;

scanf("%d", &n);

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

scanf("%d %c", &num[i], &sign[i]);

for (i = n-1; i >= 0; i--)

{

if (f <= num[i] && num[i] <= r) {

ans += num[i];

}

if (sign[i] == '+' ) {

if (num[i] < r) r = num[i];

} else { // sign[i] == '-'

if (num[i] > f) f = num[i];

}

if (f > r) break;

}

printf("%lld", ans);

return 0;

}

# 摸鱼

#include <cstdio>

#include <vector>

#include <queue>

using namespace std;

typedef long long LL;

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

// copyed\_vec means the parameter is copyed from real param,

// which makes sure we don't destroy the sequence of the original vector

LL GetCost(vector<int> copyed\_vec, int m)

{

LL ret = 0;

int w = 0;

vector<int>::iterator biter = copyed\_vec.begin();

vector<int>::iterator eiter = copyed\_vec.begin() + m;

std::make\_heap(biter, eiter);

for (int i = m; i < copyed\_vec.size(); i++)

{

w++;

ret += w \* copyed\_vec.front();

pop\_heap(biter, eiter);

copyed\_vec[m - 1] = copyed\_vec[i];

push\_heap(biter, eiter);

}

while (biter < eiter)

{

w++;

ret += w \* copyed\_vec.front();

std::pop\_heap(biter, eiter);

--eiter;

}

return ret;

}

void Solve(const vector<int>& vec, int n, LL q)

{

int lo = 1, hi = n-1, mid;

LL cost;

int tmp\_m = n;

cost = GetCost(vec, n);

if (cost > q)

{

printf("Impossible");

return;

}

else

{

printf("Possible\n");

}

while (lo <= hi) // be care for lo equals to hi

{

mid = (lo + hi) >> 1;

cost = GetCost(vec, mid);

if (cost <= q)

{

tmp\_m = min(tmp\_m, mid);

hi = mid - 1;

} else { // be care tor think about that we can't break from here

lo = mid + 1;

}

}

printf("%d", tmp\_m);

}

int main(void)

{

int n;

LL q;

scanf("%d %lld", &n, &q);

vector<int> vec(n);

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

scanf("%d", &vec[i]);

}

Solve(vec, n, q);

return 0;

}

# 智慧果分堆

#include <cstdio>

#include <iostream>

#include <functional>

#include <queue>

typedef long long LL;

void Solve(std::vector<int>& vec, int n, LL sum)

{

LL cost = 0;

int tmp = 0, cnt;

std::vector<int>::iterator biter = vec.begin();

std::vector<int>::iterator eiter = vec.end();

std::make\_heap(biter, eiter, std::greater<int>());

if (!(n & 1) && n > 3)

{

\*eiter = 0;

++eiter;

std::push\_heap(biter, eiter, std::greater<int>());

}

while (eiter - biter >= 3)

{

tmp = 0;

for (int i =0; i < 3; i++) {

std::pop\_heap(biter, eiter, std::greater<int>());

tmp += (\*--eiter);

}

\*eiter = tmp;

++eiter;

std::push\_heap(biter, eiter, std::greater<int>());

cost += tmp;

}

printf("%lld", cost);

}

int main(void)

{

int n;

LL sum = 0;

scanf("%d", &n);

std::vector<int> vec(n);

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

{

scanf("%d", &vec[i]);

sum += vec[i];

}

Solve(vec, n, sum);

return 0;

}

# 朋友圈 （并查集）

#include <cstdio>

const int N = 100005;

int par[N];

int cnt[N];

int Find(int x)

{

if (par[x] == -1) return x;

return par[x] = Find(par[x]);

}

void Union(int x, int y)

{

int u = Find(x);

int v = Find(y);

if (u == v) return ; // important

par[u] = v;

cnt[v] += cnt[u];

cnt[u] = 0;

}

int main(void)

{

int n, m;

scanf("%d %d", &n, &m);

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

{

par[i] = -1;

cnt[i] = 1;

}

int x, y;

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

{

scanf("%d %d", &x, &y);

Union(x, y);

}

int max = cnt[0];

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

if (cnt[i] > max) max = cnt[i];

printf("%d", max);

return 0;

}

# 破碎的数组 (并查集 + offline)

#include <cstdio>

#include <stack>

using std::stack;

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

typedef long long ll;

const int maxn = 100000 + 5;

int fa[maxn] = { 0 };

ll sum[maxn] = { 0 };

int del[maxn] = { 0 };

bool vis[maxn] = { false };

ll ans = 0;

int Find(int x)

{

if (fa[x] == 0) return x;

return fa[x] = Find(fa[x]);

}

void Union(int x, int y)

{

int u, v;

u = Find(x);

v = Find(y);

if (u == v) return ;

fa[u] = v;

sum[v] += sum[u];

sum[u] = 0;

}

void Solve(int n)

{

ll ans = 0;

int cur;

stack<ll> st;

st.push(0);

for (int i = n; i > 1; i--)

{

cur = del[i];

vis[cur] = true;

if (cur == n && vis[n - 1])

{

Union(n - 1, n);

}

else

{

if (vis[cur - 1])

Union(cur - 1, cur);

if (vis[cur + 1])

Union(cur, cur + 1);

}

ans = max(ans, sum[Find(cur)]);

st.push(ans);

}

while (!st.empty())

{

printf("%lld\n", st.top());

st.pop();

}

}

int main(void)

{

int n;

scanf("%d", &n);

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

scanf("%d", &sum[i]);

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

scanf("%d", &del[i]);

Solve(n);

return 0;

}

# 最短路径 (forward star)

#include <cstdio>

#include <cstring>

#include <queue>

typedef long long ll;

const int INF = 0x3f3f3f3f;

const int V = 50000 + 5; // max vertex number

const int E = 200000 + 5;// max edge number

// Chain-forward-star

int n, m, ecnt = 0, head[V] = { 0 };

struct node

{

int v, next, w;

} edge[E \* 2];

void AddEdge(int u, int v, int w)

{

edge[++ecnt].v = v;

edge[ecnt].w = w;

edge[ecnt].next = head[u];

head[u] = ecnt;

}

std::priority\_queue<std::pair<ll, int> > Q;

ll dis[V];

bool vis[V] = { false };

void Dijkstra()

{

memset(dis, 0x3f, sizeof(dis));

dis[1] = 0;

Q.push(std::make\_pair(0, 1)); // weight + number

while (!Q.empty())

{

int u = Q.top().second; // from V-S pick the vertex with the min distance

Q.pop();

if (vis[u])

continue;

vis[u] = true; // add to S

for (int i = head[u]; i; i = edge[i].next)

{ // enumerate the edges linked to u

int v = edge[i].v, w = edge[i].w;

if (dis[v] > dis[u] + w)

{ // search for edge which can loose

dis[v] = dis[u] + w; // update distance

Q.push(std::make\_pair(-dis[v], v));

}

}

}

return ;

}

int main()

{

scanf("%d %d", &n, &m);

int a, b, c;

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

{

scanf("%d %d %d", &a, &b, &c);

AddEdge(a, b, c);

AddEdge(b, a, c); //无向图需要存两遍

}

Dijkstra();

printf("%d", dis[n]);

return 0;

}

# 最短路径2 （邻接表）

#include <cstdio>

#include <algorithm>

#include <queue>

using namespace std;

#define swap(a, b) do { int t = a; a = b; b = t; } while (0)

const int V = 50005; /\*, E = 200005\*/

const int INF = 0x3f3f3f3f;

struct edge { int to, cost; };

typedef pair<int, int> P; // first => distance, second => vertex number

vector<edge> G[V];

int d[V];

priority\_queue<P, vector<P>, greater<P> > Q;

void Dijkstra(int s)

{

memset(d, INF, sizeof(d));

d[s] = 0;

Q.push(P(0, s));

int v;

edge e;

while (!Q.empty())

{

P p = Q.top(); Q.pop();

v = p.second;

if (d[v] < p.first) continue;

for (int i = 0; i < G[v].size(); ++i) {

e = G[v][i];

if (d[e.to] > d[v] + e.cost) {

d[e.to] = d[v] + e.cost;

Q.push(P(d[e.to], e.to));

}

}

}

}

int main(void)

{

int n, m;

scanf("%d %d", &n, &m);

int a;

edge e;

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

{

scanf("%d %d %d", &a, &e.to, &e.cost);

G[a].push\_back(e);

swap(a, e.to); // swap src and dest

G[a].push\_back(e);

}

Dijkstra(1);

printf("%d", d[n]);

return 0;

}

# 贪心大法官 （拓扑排序）

#include <iostream>

#include <algorithm>

#include <vector>

#include <queue>

using namespace std;

const int N = 10000 + 5;

vector<int> G[N]; // graph

priority\_queue<int, vector<int>, greater<int> > Q;

int in[N] = { 0 }; // indegree

void TopSort(int n)

{

int v;

for (v = 1; v <= n; v++)

{

if (in[v] == 0)

Q.push(v);

}

while (!Q.empty())

{

v = Q.top();

Q.pop();

cout << v << " ";

// for (vector<int>::iterator it = G[v].begin(); it != G[v].end(); ++it)

for (int i = 0; i < G[v].size(); i++)

{

int after = G[v][i];

if (--in[after] == 0)

Q.push(after);

}

}

}

int main()

{

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

cin.tie(NULL);

int n, m, t;

cin >> n;

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

{

cin >> m;

for (int j = 0; j < m; j++)

{

cin >> t;

Q.push(t);

in[t]++;

}

while (!Q.empty())

{

t = Q.top();

Q.pop();

G[i + 1].push\_back(t);

}

}

TopSort(n);

return 0;

}

# 二叉树遍历

#include <stdio.h>

const int N = 10000 + 5;

int pre[N], in[N];

int mapIndex[N]; // map for getting one element from inorder in O(1)

void PostOrder(int root, int begin, int end)

{

if (begin > end) return ; // End the recursion

int rootVal = pre[root];

int mid = mapIndex[rootVal];

PostOrder(root+1, begin, mid-1); // left-child

PostOrder(root+1+mid-begin, mid+1, end); // right-child

printf("%d ", rootVal); // root

}

void PreOrder(int root, int front, int end)

{

if (front > end) return ;

int rootVal = post[root];

int mid = mapIndex[rootVal];

printf("%d ", rootVal); // root

PreOrder(root-1-end+mid, front, mid-1); // left-child

PreOrder(root-1, mid+1, end); // right-child

}

int main(void)

{

int n, i;

scanf("%d", &n);

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

scanf("%d", &pre[i]);

}

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

scanf("%d", &in[i]);

mapIndex[in[i]] = i;

}

PostOrder(0, 0, n-1);

return 0;

}

# 找某个字符串的不同子串的数目(O(n^2))

#include <iostream>

#include <string>

#include <set>

using namespace std;

unsigned Solve(const string& str)

{

set<string> st;

st.clear();

for (size\_t i=0; i<str.size(); i++)

for (size\_t j=1; i+j<=str.size(); j++)

st.insert(str.substr(i, j));

return st.size();

}

int main(void)

{

string str;

cin >> str;

cout << Solve(str);

return 0;

}

# 二分查找

#include <stdio.h>

typedef long long ll;

int a[100005];

// the work capacity of the machine

ll Capacity(int n, ll mid, int m)

{

ll ans = 0;

int i;

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

{

ans += mid / a[i];

if (ans > m) // have washed enough clothes

return ans;

}

return ans;

}

int main()

{

int n, m, i;

ll ans = 0;

ll right, left, mid, now;

scanf ("%d %d", &n, &m);

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

scanf ("%d",&a[i]);

left = 0;

right = 1e11; // 100000 \* 1000000

while (right >= left)

{

mid = (right + left) >> 1;

ans = Capacity(n, mid, m);

if (ans > m)

{

now = mid;

right = mid - 1;

}

else if (ans < m)

{

left = mid + 1;

}

else// (ans==m)

{

now = mid;

break;

}

}

printf ("%lld", now);

return 0;

}

# 最小生成树（KRUSKAL）

#include <cstring>

#include <iostream>

#include <algorithm>

using namespace std;

const int E = 1000;

int par[E];

struct Edge

{

int u, v, w;

bool operator<(const Edge& e) const

{

return this->w < e.w;

}

} edge[E << 1]; // 无向图

int ecnt = 0;

void AddEdge(int u, int v, int w)

{

ecnt++;

edge[ecnt].u = u;

edge[ecnt].v = v;

edge[ecnt].w = w;

}

int Find(int x)

{

if (par[x] == -1) return x;

return par[x] = Find(par[x]);

}

bool Union(int x, int y)

{

x = Find(x);

y = Find(y);

if (x == y) return false;

par[x] = y;

return true;

}

int Kruskal(int n)

{

memset(par, -1, sizeof(par));

sort(edge + 1, edge + ecnt + 1); // [1..ecnt]

int k, s, t;

int ans = 0;

for (int i = 1; i <= ecnt && k < n-1; i++)

{

if (Union(edge[i].u, edge[i].v))

{

ans += edge[i].w;

k++;

}

}

return ans;

}

int main()

{

int n;

char from, to;

int e, w;

while (cin >> n && n != 0)

{

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

{

cin >> from >> e;

for (int j = 0; j < e; j++)

{

cin >> to >> w;

AddEdge(from, to, w);

AddEdge(to, from, w); // 无向图

}

}

cout << Kruskal(n) << endl;

ecnt = 0;

}

return 0;

}

#include <cstdio>

#include <cstring>

#include <queue>

using namespace std;

const int N = 100005;

typedef long long ll;

struct Edge

{

int src, dest, cost;

bool operator < (const Edge& o) const {

return this->cost > o.cost; // used in priority\_queue(min heap)

}

};

priority\_queue<Edge> Q;

Edge E;

// UFSet

int par[N];

int Find(int x)

{

if (par[x] == 0) return x;

return par[x] = Find(par[x]);

}

void Union(int x, int y)

{

x = Find(x);

y = Find(y);

if (x != y) par[x] = y;

}

ll Kruskal(int n, int m)

{

ll ans = 0;

int k = 0;

int s, t;

for (int i = 0; i < m && k < n - 1; i++)

{

E = Q.top();

Q.pop();

s = Find(E.src);

t = Find(E.dest);

if (s != t) {

Union(s, t);

ans += E.cost;

}

}

return ans;

}

int main()

{

int n, m;

scanf("%d %d", &n, &m);

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

scanf("%d %d %d", &E.src, &E.dest, &E.cost);

Q.push(E);

}

printf("%d", Kruskal(n, m));

return 0;

}