23级算法设计与分析期末考试部分题解

> 22375080 - DeNeRATe 杨佳宇轩

A

题目简述

输入一个数 n，输出 `I promise ...` 字符串 n 次

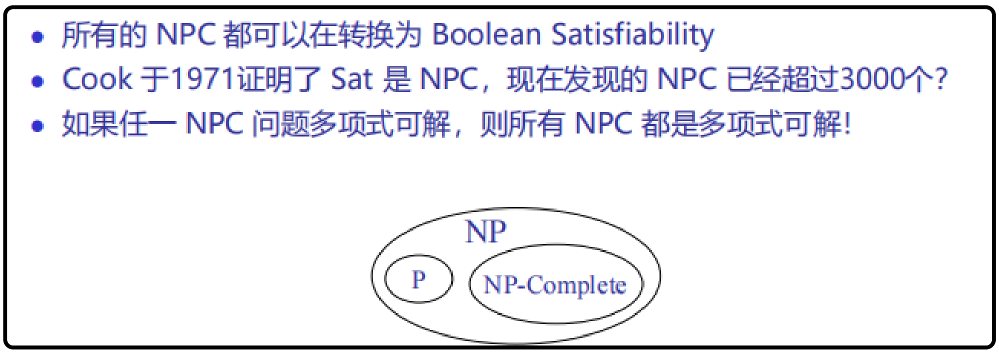
题目分析

==诚信题==（但竟然需要一个循环，还是太不签到了（据说前年是复制代码并提交）

B

题目简述（部分选择题）

- NP问题与P问题的关系



- 动态规划和贪心都是最优子结构
- KMP的 π 函数运行结果（`ababa`）
- 时间复杂度排序 $n\log n, n\log^2 n, n!, n^{\frac{2}{3}}\cdots$
- 主方法时间复杂度计算

$$T(n) = aT(n/b) + f(n),$$

$$T(n) = \begin{cases} \Theta\left(n^{\log_b a}\right), & f(n) = O\left(n^{(\log_b a)-\varepsilon}\right) \\ \Theta\left(n^{\log_b a}\lg n\right), & f(n) = \Theta\left(n^{\log_b a}\right) \\ \Theta(f(n)), & f(n) = \Omega\left(n^{(\log_b a)+\varepsilon}\right) \text{ and } af(n/b) \le cf(n) \text{ for large } n \end{cases} \quad \exists \varepsilon > 0, c < 1$$

C

题目简述

给定 n 个数，输出其中的奇数个数和偶数个数

题目分析

直接遍历即可

示例代码

```
for(int i = 1; i <= n; i++) {
    int x; cin >> x;
    if(x % 2 == 0) cnt[0]++;
    else cnt[1]++;
}
```

D

题目简述

给定一个区间 $[l, r]$，求其中是 x 的倍数的个数

$r - l + 1 \le 10^7$

题目分析

由区间范围可以知道直接进行遍历判断即可

示例代码

```
for(int i = l; i <= r; i++) {
    if(i % x == 0) cnt++;
}
```

E

题目简述

给定 n 个数 $a_1, \cdots, a_n$，给出 m，之后不断在数列中加入随机数，问至少加入多少次可以使得 a_m 称为数列的第 n 小

题目分析

最优情况即每次加入的随机数都是 $< a_m$，可以使得 a_m 的排序位置变化最快

所以我们先统计数列中 $< a_m$ 的数的个数，设为 x，则最后只需要最少加入 $n - x - 1$ 个即可

示例代码

```
vector<int> a(n);
rep(i, 0, n - 1) cin >> a[i];
int lim = a[m - 1], x = 0;
rep(i, 0, n - 1) x += (a[i] < lim);
cout << n - x - 1 << endl;
```

F

题目简述

n 个晾衣杆，其长度为 l_i，其中已经晾了 k_i 件衣服，每个晾衣杆有一个宽敞度定义为 $\frac{l_i}{k_i+1}$，之后还会晾 m 件衣服，每次都会晾在宽敞度最高的晾衣杆上，如果宽敞度相同，则放在编号最小的晾衣杆上边

题目分析

易知，这一个比较过程是一个**线序**，每次加入一个元素都只需要找最优的那一个，所以可以直接优先队列进行修改推入等操作

但注意，宽敞度的比较由于涉及分数计算以及比较，所以应该使用交叉相乘进行比较：i 优于 $j \Leftrightarrow l_i \times (k_j + 1) > l_j \times (k_i + 1)$

时间复杂度：$O(n \log n)$

示例代码

```cpp
struct node {
    int l, k, id;
    friend bool operator < (node a, node b) {
        if(a.l * (b.k + 1) != b.l * (a.k + 1))
            return a.l * (b.k + 1) < b.l * (a.k + 1);
        else return a.id > b.id;
    }
};

...

inline void SOLVE() {
    priority_queue<node> q;
    rep(i, 1, n) {
        int l, k;
        cin >> l >> k;
        q.push(node{l, k, i});
    }
    while(t--) {
        node x = q.top();
        q.pop();
        x.k++;
        cout << x.id << endl;
        q.push(x);
    }
}
```

G

题目简述

给定一个**正整数**序列，从中选出若干连续段，保证每一连续段的长度不超过 k，其价值为所有选中的数之和，求最大为多少

其中 $n \sim 10^5, k \leq 10$

题目分析

一眼dp题，对于题目要求，我们需要关注的其实是一个连续段的最后一个元素的位置，因此对于 `f[i]` 我们定义为最后一个选择的位置 $\le i$ 且满足题目要求的最大价值和

由于涉及区间和，所以可以想到加上前缀和优化

转移方程：

有两种转移方式：

- 这一个位置不选，那么直接继承上一个位置的值
- 选这个位置，那么枚举所有 **间隔** 的位置进行转移

$$f[i] = max_{j=i-k-1}^{i-2}\{f[j] + val[j+2 \sim i]\}$$
$$f[i] = max\{f[i-1], f[i]\}$$

示例代码

```
vector<int> a(n + 1), pre(n + 1);
rep(i, 1, n) cin >> a[i], pre[i] = pre[i - 1] + a[i];
vector<int> f(n + 1);
rep(i, 1, n) {
    f[i] = f[i - 1];
    rep(j, max(0, i - k - 1), i - 2) {
        f[i] = max(f[i], f[j] + pre[i] - pre[j + 1]);
    }
}
cout << f[n] << endl;
```

H

题目简述

给定一个长度为 n 的序列，每次删除两个位置 i, j 当且仅当 $[i+1, j-1]$ 中所有的元素都已经被删除，每次删除都有一个费用 f_{ij}，多次删除的费用为所用到删除费用的最大值，问删除整个序列的最小费用

$n \le 800$

题目分析

看到数据范围 + 区间类型的操作，可以直接想到是区间dp了

我们定义 $f[i][j]$ 为删除完了区间 $[i, j]$ 所用的最小费用

注意，每次操作的区间一定是一个**偶数长度**

有两种转移方式：

- 最后一次删除 i,j ，则 $f[i][j]=f[i+1][j-1]+val[i][j]$
- 删除区间包含多个区间，虽然是多个区间，但其实可以看作两个区间的并，所以只需要枚举断点 $f[i][j]=min_{k=i+1}^{j-2}\{max(f[i][k],f[k+1][j])\}$

时间复杂度：$O(n^3)$

示例代码

```
rep(i, 1, n - 1)
    f[i][i + 1] = val[i][i + 1];
for(int len = 4; len <= n; len += 2) {
    rep(l, 1, n - len + 1) {
        int r = l + len - 1;
        f[l][r] = f[l + 1][r - 1] + val[l][r];
        for(int k = l + 1; k <= r - 2; k += 2)
            f[l][r] = min(f[l][r], max(f[l][k], f[k + 1][r]));
    }
}
cout << f[1][n] << endl;
```

I

题目简述

总共有 n 个同学，一名助教，编号为 $n+1$，若干人之间相互认识，某些同学也和助教相互认识，相互认识的人之间可以传递作业，作业传递不限数量，全班都可以同时传递，问助教最早多久才可以收满作业

题目分析

与期末考试前的模拟考试很像，但那道题是最小生成树，这道题是**单源最短路**！

因为我们只需要关注和助教距离最远的即可

通过Dijkstra计算出所有点到 $n+1$ 的最短路，枚举所有点取最小值即可

时间复杂度：$(n+m)\log n$

示例代码

```cpp
/* --- Dijkstra板子 --- */
inline vll dijkstra(int sta, int n, vvpii &edge) {
    vll dis(n, 0x3f3f3f3f3f3f3f3f); dis[sta] = 0;
    vb jud(n);
    priority_queue<pair<ll, int>, vector<pair<ll, int>>,
greater<pair<ll, int>>> q;
    q.push(pll{dis[sta], sta});
    while(!q.empty()) {
        pair<ll, int> now = q.top();
        q.pop();
        if(jud[now.second]) continue;
        jud[now.second] = true;
        for(auto &[ed, val]: edge[now.second]) {
            if(dis[ed] > now.first + val) {
                dis[ed] = now.first + val;
                q.push(make_pair(dis[ed], ed));
            }
        }
    }
    return dis;
}

/* --- 建图 --- */
inline void SOLVE() {
    vvpii e(n + 1);
    rep(i, 1, m) {
        int u, v, w;
        cin >> u >> v >> w;
        u--, v--;
        e[u].push_back({v, w});
        e[v].push_back({u, w});
    }
    vll dist = dijkstra(n, n + 1, e);
    ll ans = 0;
    rep(i, 0, n) ans = max(ans, dist[i]);
    cout << ans << endl;
}
```

J

题目简述

安排考场，n 名学生，m 间考场，每名学生可以去若干间考场考试，每个考场有一定的容量，求是否存在一个方法使得每位考生都可以在考场考试

题目分析

二分图最大匹配问题，可以使用最大流解决

建立超级源点 S 和超级汇点 T

- 对于每位考生 i，赋予 **1** 的流量，$S \to i$
- 对于考生 i 可以选择的考场 c_{ij}，赋予 **1** 的流量，$i \to c_{ij}$
- 对于第 i 考场容量 w_i，赋予 w_i 流量，$i \to T$，即最多可以有 w_i 个学生流经这间教室

最后直接跑一边最大流即可

时间复杂度：$O(n^2m) \parallel O(n^2\sqrt{m})$

> 本人 Dinic 考场板子挂了，最后也没有改成预流推进，自闭……

示例代码

将 **0** 作为超级源点，$n+m+1$ 作为超级汇点

此处放可以过的 `PushRelabel` 代码，~~就不放挂了的 `Dinic` 了，呜呜呜~~

```
struct PushRelabel {
    struct Edge {
        int ed, rev;
        ll f, c;
    };

    vc<vc<Edge>> g; // graph
    vll ec; // excess flow
    vc<Edge*> cur;
    vvi hs; // bucket for height
    vi H; // height

    PushRelabel(int n): g(n), ec(n), cur(n), hs(n << 1), H(n)
{}

    void addEdge(int s, int t, ll cap, ll rcap = 0) {
        if(s == t) return;
        g[s].push_back({t, sz(g[t]), 0, cap});
        g[t].push_back({s, sz(g[s]) - 1, 0, rcap});
    }

```

```cpp
    void addFlow(Edge& e, ll f) {
        Edge &rev = g[e.ed][e.rev];
        if(!ec[e.ed] && f) { hs[H[e.ed]].push_back(e.ed); }
        e.f += f; e.c -= f; ec[e.ed] += f;
        rev.f -= f; rev.c += f; ec[rev.ed] -= f;
    }

    ll calc(int s, int t) {
        int v = sz(g); H[s] = v; ec[t] = 1;
        vi cop(v << 1); cop[0] = v - 1;
        rep(v) cur[i] = g[i].data();
        for(Edge& e: g[s]) addFlow(e, e.c);

        for(int hi = 0;;) {
            while(hs[hi].empty()) if(!hi--) return -ec[s];
            int u = hs[hi].back(); hs[hi].pop_back();
            while(ec[u] > 0)
                if(cur[u] == g[u].data() + sz(g[u])) {
                    H[u] = 1e9;
                    for(Edge& e: g[u]) if(e.c && H[u] >
H[e.ed] + 1)
                        H[u] = H[e.ed] + 1, cur[u] = &e;
                    if(++cop[H[u]], !--cop[hi] && hi < v)
                        rep(v) if(hi < H[i] && H[i] < v)
                            --cop[H[i]], H[i] = v + 1;
                    hi = H[u];
                } else if(cur[u] -> c && H[u] == H[cur[u] ->
ed] + 1)
                    addFlow(*cur[u], min(ec[u], cur[u] -> c));
                else ++cur[u];
        }
    }
};

inline void SOLVE() {
    INT(n, m);
    PushRelabel a(n + m + 2);
    rep(i, 1, m) {
        int x; cin >> x;
        a.addEdge(i + n, n + m + 1, x);
    }
    rep(i, 1, n) {
        int c; cin >> c;
        rep(j, 1, c) {
            int x; cin >> x;
            a.addEdge(i, x + n, 1);
        }
        a.addEdge(0, i, 1);
    }
    cout << a.calc(0, n + m + 1) << endl;
}
```

K

【没看，应该是FFT】

L

题目简述

给定一个字符串 s 以及多个模式串 t_i，求可以和 s 匹配最多次数的模式串，若匹配次数相同则输出编号最小的

题目分析

KMP模板题，直接暴力每一个模式串 t_i 并求得匹配位置个数进行比较即可

示例代码

```
vi getLost(const string &s) {
    vi p(sz(s));
    rep(i, 1, sz(s) - 1) {
        int g = p[i - 1];
        while (g && s[i] != s[g])
            g = p[g - 1];
        p[i] = g + (s[i] == s[g]);
    }
    return p;
}

vi match(const string &s, const string &pat) {
    vi p = getLost(pat + '\0' + s), res;
    rep(i, sz(p) - sz(s), sz(p) - 1)
        if (p[i] == sz(pat))
            res.push_back(i - 2 * sz(pat));
    return res;
}

inline void SOLVE() {
    INT(n);
    str s; cin >> s;
    int mx = -1, id = -1;
    vs t(n);
    rep(n) {
```

```cpp
        cin >> t[i];
        vi res = match(s, t[i]);
        if(sz(res) > mx)
            mx = sz(res), id = i;
    }
    cout << t[id] << endl;
}
```

M

【没看】