CSP 202203 实验报告

学号：2022212408 姓名：胡宇杭

2023 年 5 月 14 日

提交结果

3674684	胡宇杭	胡宇杭	未初始化警告	05-13 23:16	467B	CPP14	正确	100	140ms	3.027MB
3674683	胡宇杭	胡宇杭	出行计划	05-13 23:15	568B	CPP14	正确	100	312ms	4.546MB
3674682	胡宇杭	胡宇杭	计算资源调度器	05-13 23:15	2.516KB	CPP14	正确	100	578ms	3.335MB
3674681	胡宇杭	胡宇杭	通信系统管理	05-13 23:14	2.392KB	CPP14	正确	100	1.937s	50.70MB
3674670	胡宇杭	胡宇杭	博弈论与石子合并	05-13 23:10	1.042KB	CPP14	正确	100	78ms	4.035MB

图 1: 提交结果

1 202203-1 未初始化警告

- 时间限制: 1.0 s
- 空间限制: 512.0 MB

1.1 题目

1.1.1 问题描述

考虑一段包含 k 条赋值语句的简单代码。该段代码最多使用到 n 个变量，分别记作 $a_1, a_2, \cdots, a_n$；常量用 a_0 表示。

对于任意一条赋值语句 $a_{x_i} = a_{y_i}$，如果右值 a_{y_i} 是一个变量，则其应该在此之前被初始化过，否则认为该条语句存在右值未初始化的问题。

按照上述规则，试统计给定的代码中，有多少条赋值语句右值未被初始化。

1.1.2 子任务

50% 的测试数据满足 $0 < n, k \leq 1000$；
全部的测试数据满足 $0 < n, k \leq 10^5$。

1.2 题解

1.2.1 解法一 (100 分)

在一条赋值语句中，无论右值是否被初始化过，左值都会被初始化，因此可以定义 $initial$ 数组，对于每一个变量 i，$initial[i]$ 记录该变量是否被初始化过。初始时将 $initial[0]$ 赋值为真，表示常数。

- 时间复杂度: $O(k)$
- 空间复杂度: $O(n)$

1.2.2 C++ 代码实现

202203-1 未初始化警告

```cpp
#include <iostream>
#include <algorithm>
#include <cstring>
using namespace std;
#define maxn 100005

int n, k, op1, op2, ans = 0;
bool initial[maxn];

int main()
{
    cin >> n >> k;

    initial[0] = true;
    for (int i = 1; i <= k; i++)
    {
        cin >> op1 >> op2;
        if (initial[op2] == true)
            initial[op1] = true;
```

```
20          else
21              ans++, initial[op1] = true;
22      }
23
24      cout << ans << endl;
25
26      // system("pause");
27
28      return 0;
29  }
```

2 202203-2 出行计划

- 时间限制：1.5 s
- 空间限制：512.0 MB

2.1 题目

2.1.1 问题描述

在某个神奇的地方，出入各个场所都要持有一定时间内的核酸检测阴性证明。若在 t 时刻做了核酸，在等待 k 时间后可以获得核酸证明。更具体地，如果一个场所要求 c 个时间内核酸证明，则可以在第 $t+k$ 时刻到第 $t+k+c-1$ 内进入该场所。

按照上述规则，给出每个场所的访问时间和要求核酸证明的时间，若 q 时间做了核酸，则一共有多少项出行计划的核酸要求可以得到满足。

2.1.2 子任务

40% 的测试数据满足 $0 < n, k \leq 1000$、$m = 1$；
70% 的测试数据满足 $0 < n, m, k \leq 1000$；
全部的测试数据满足 $0 < n, m, k \leq 10^5$。

2.2 题解

2.2.1 解法一 (70 分)

用两个数组 t、c 记录每个场所的访问时间和要求核酸证明的时间，对于每次询问，依次遍历每个场所，若 $t+k \le t[i] \le t+k+c[i]-1$，则 ans++。可通过 70% 的数据。

- 时间复杂度：$O(nm)$
- 空间复杂度：$O(n)$

2.2.2 C++ 代码实现

202203-2 出行计划

```cpp
#include <iostream>
#include <algorithm>
#include <cstring>
#include <cmath>
using namespace std;
#define maxn 100005

int n, m, k, q, t[maxn], c[maxn];

int main()
{
    cin >> n >> m >> k;

    for (int i = 1; i <= n; i++)
        cin >> t[i] >> c[i];

    for (int i = 1; i <= m; i++)
    {
        cin >> q;
        int l = q + k, ans = 0;
        for (int j = 1; j <= n; j++)
        {
            int r = l + c[j] - 1;
            if (l <= t[j] && t[j] <= r)
                ans++;
```

```
        }
        cout << ans << endl;
    }

    // system("pause");

    return 0;
}
```

2.2.3 解法二 (100 分)

本题涉及到对区间的查询，可以考虑使用差分对数据进行处理，对每一个场所，如果访问时间为 t，则只要 $q(t-c+1 \le q \le t)$ 时刻持有核酸证明，就可以访问该场所。因此，可以定义数组 a, 令 $a[t-c+1]+1$，$a[t+1]-1$，再对 a 求前缀和 sum 。查询时只要访问 $sum[q+k]$ 处的元素即可。

- 时间复杂度：$O(n+m)$

202203-2 出行计划

```
#include <iostream>
#include <algorithm>
#include <cstring>
#include <cmath>
using namespace std;
#define maxn 200100

int n, m, k, q, t, c, a[maxn], sum[maxn];

int main()
{
    cin >> n >> m >> k;

    for (int i = 1; i <= n; i++)
    {
        cin >> t >> c;
        int temp = t - c + 1;
        a[temp > 0 ? temp : 1]++;
        a[t + 1]--;
    }
```

```
21
22      for (int i = 1; i <= 200000; i++)
23          sum[i] = sum[i - 1] + a[i];
24
25      for (int i = 1; i <= m; i++)
26      {
27          cin >> q;
28          cout << sum[q + k] << endl;
29      }
30
31      // system("pause");
32
33      return 0;
34  }
```

3 202203-3 计算资源调度器

- 时间限制：1.0 s
- 空间限制：512.0 MB

3.1 题目

3.1.1 问题描述

一个云计算平台可分为多个可用区，每个计算节点位于一个特定的可用区，一个可用区中可以有多个计算节点。每个计算任务都有与之对应的应用，且都执行在特定节点上。

现需要按照给定要求对计算节点进行分配，要求如下：

- 计算节点亲和性：计算任务必须要运行在指定可用区上。
- 计算任务亲和性：计算任务必须要和指定应用的计算任务运行在同一可用区上。
- 计算任务反亲和性：计算任务不可以和指定应用的计算任务运行在同一可用区上。无可用区时，若要求为尽量满足，则忽略反亲和性要求，否则认为该计算任务无法分配。

按照上述要求对计算节点进行筛选后，对剩余节点进行排序

- 优先选择此时运行计算任务最少的计算节点。
- 若运行计算任务数量相同，优先选择编号小的计算节点。

3.1.2 子任务

本题包含 20 个测试用例，每个 5 分。
全部测试数据保证 $0 < m \leq n \leq 1000$，$0 < g \leq \sum_{n=1}^{g} f_i \leq 2000$。
部分测试点的特殊性质详见下表。

测试点	最大应用编号	计算节点亲和性	计算任务亲和性	计算任务反亲和性
1，2	$A_{max} = 10$	无要求（$na_i = 0$）	无要求（$pa_i = 0$）	无要求（$paa_i = 0$）
3，4	$A_{max} = 10^9$	无要求（$na_i = 0$）	无要求（$pa_i = 0$）	无要求（$paa_i = 0$）
5，6，7	$A_{max} = 10$	$na_i \geq 0$	无要求（$pa_i = 0$）	无要求（$paa_i = 0$）
8，9，10	$A_{max} = 10^9$	$na_i \geq 0$	无要求（$pa_i = 0$）	无要求（$paa_i = 0$）
11，12，13	$A_{max} = 10$	$na_i \geq 0$	无要求（$pa_i = 0$）	$paa_i \geq 0$
14，15，16	$A_{max} = 10^9$	$na_i \geq 0$	无要求（$pa_i = 0$）	$paa_i \geq 0$
17，18	$A_{max} = 10$	$na_i \geq 0$	$pa_i \geq 0$	$paa_i \geq 0$
19，20	$A_{max} = 10^9$	$na_i \geq 0$	$pa_i \geq 0$	$paa_i \geq 0$"]

图 2: 测试数据

3.2 题解

3.2.1 解法一 (100 分)

定义结构体 $Node$ 存放每个计算节点的编号，所在可用区，运行计算任务的数量，同时定义 $set<int>\ task$ 存放当前节点运行的计算任务的编号。定义 $vector<int>\ node$ 存放 $Node$。在筛选过程中，定义 $vector<int>\ temp$ 用于筛选计算节点。筛选完后根据题意使用 $std::sort$ 对剩余节点进行排序，取首元素并对相应 $Node$ 的状态进行更新。

具体筛选方法如下：

- 计算节点亲和性：遍历 $temp$，若 $node[i].block! = na$，则删除该计算节点。
- 计算任务亲和性:定义 $set<int>\ temp\_s$,首先遍历 $node$ 数组,若 $node[i].task$ 中存放有对应计算任务，则将其 $block$ 插入 $temp\_s$ 中，接下来遍历 $temp$ 数组，若 $node[i].block$ 在 $temp\_s$ 中未出现，则删除该计算节点。
- 计算任务反亲和性：定义数组 $temp2$, 将经过上述程序筛选后的 $temp$ 赋值给 $temp2$。遍历 $temp2$，若 $node[i].task$ 中存放有对应计算任务，则删除该计算节点。
- 对于筛选完的 $temp\ temp2$，若 $temp2$ 为空并且 $paar = 1$，或 $temp$ 为空，则该计算任务无法分配。

需要注意的是，如果直接定义 cmp 函数使用 $std::sort$ 进行自定义排序会 TLE，只能通过 80% 的数据。因此这里采用 $lambda$ 表达式实现，可通过 100% 的数据。

- 时间复杂度：$O(n^2logn)$

3.2.2 C++ 代码实现

202203-3 计算资源调度器

```cpp
#include <iostream>
#include <algorithm>
#include <cstring>
#include <vector>
```

```cpp
#include <set>
#include <unordered_map>
using namespace std;

struct Node
{
    int id, block, cnt;
    set<int> task;
    bool operator<(const Node &a) const
    {
        if (this->cnt == a.cnt)
            return this->id < a.id;
        return this->cnt < a.cnt;
    }
};

unordered_map<int, Node> node;
vector<int> v;
int n, m, g, l, f, a, na, pa, paa, paar;

int main()
{
    cin >> n >> m;

    for (int i = 1; i <= n; i++)
    {
        cin >> l;
        set<int> temp;
        node.insert(make_pair(i, Node{i, l, 0, temp}));
        v.push_back(i);
    }

    cin >> g;
    for (int i = 1; i <= g; i++)
    {
        cin >> f >> a >> na >> pa >> paa >> paar;
        for (int j = 1; j <= f; j++)
        {
            vector<int> temp = v;
            if (na)
            {
                for (auto it = temp.begin(); it != temp.end();)
```

```
47                  {
48                      if (node[*it].block != na)
49                          temp.erase(it);
50                      else
51                          it++;
52                  }
53              }
54
55              if (pa)
56              {
57                  set<int> temp_s;
58                  for (auto &it : node)
59                  {
60                      if (it.second.task.count(pa) == 1)
61                      {
62                          temp_s.insert(it.second.block);
63                      }
64                  }
65
66                  for (auto it = temp.begin(); it != temp.end();)
67                  {
68                      if (!temp_s.count(node[*it].block))
69                          temp.erase(it);
70                      else
71                          it++;
72                  }
73              }
74
75              vector<int> temp2 = temp;
76
77              if (paa)
78              {
79                  for (auto it = temp2.begin(); it != temp2.end();)
80                  {
81                      if (node[*it].task.count(paa))
82                          temp2.erase(it);
83                      else
84                          it++;
85                  }
86              }
87
88              if ((temp2.empty() && paar) || temp.empty())
```

```
89              {
90                  cout << 0 << " ";
91                  continue;
92              }
93              else if (temp2.empty() && !paar)
94                  temp2 = temp;
95
96              sort(temp2.begin(), temp2.end(), [](const auto &a, const auto &b) -> bool
97                   { return node[a] < node[b]; });
98              node[temp2[0]].cnt++;
99              node[temp2[0]].task.insert(a);
100             cout << node[temp2[0]].id << " ";
101         }
102         cout << endl;
103     }
104
105     // system("pause");
106
107     return 0;
108 }
```

4 202203-4 通信系统管理

- 时间限制：2.0 s
- 空间限制：512.0 MB

4.1 题目

4.1.1 问题描述

有 n 台计算机，互相之间可以发送数据。任意两台机器之间每日可以互相发送的数据量有限（每日可用额度），并且有如下定义：

- 主要通信对象：每台机器的通信主要对象为当前时刻与该机器的每日可用额度最大的机器（并列取编号最小）。若其为 0，则该机器无主要通信对象。
- 通信孤岛：若一台机器与任何机器之间的每日可用额度均为 0，则称为通信孤岛，并认为其无主要通信对象。

- 通信对：若两台机器之间互为主要通信对象，则称为通信对。

在 $1 \sim m$ 时间内，每天都要处理若干额度申请，每个申请如 $u\ v\ x\ y$ 格式，表示机器 $u\ v$ 之间每日可用额度增大 x，持续 y 天；而后，需要查询若干机器的主要通讯对象；最后，可能需要查询此时刻的通信孤岛和通信对个数。

4.1.2 子任务

设 $A = \sum_{i=1}^{m} k_i$，$B = \sum_{i=1}^{m} l_i$。

子任务 1 (20 分)：满足 $n, m \leq 1000$，$A, B \leq 2000$；

子任务 2 (10 分)：满足 $p_i = q_i = 0$；

子任务 3 (10 分)：满足 $l_i = q_i = 0$；

子任务 4 (15 分)：满足 $l_i = 0$；

子任务 5 (10 分)：满足 $k_1 = A$，对于所有额度申请均满足 $y = m$；

子任务 6 (15 分)：满足对于所有额度申请均满足 $y = m$；

子任务 7 (20 分): 无特殊性质；

对于 100% 的数据，$1 \leq n, m \leq 10^5$，$1 \leq A, B \leq 2 \times 10^5$，$1 \leq u, v \leq n$，$1 \leq x \leq 10^9$，$1 \leq y \leq m$。

所有额度申请均满足 $u \neq v$。

4.2 题解

4.2.1 解法一 (100 分)

考虑使用邻接表存储信息。同时，因为题目需要查询某个机器的通信主要对象，可以定义结构体 $Node$，并重载 $<$ 运算符，使用 set 容器维护 $Node$。同时，用 map 映射机器 $u\ v$ 之间的可用额度。在每一时刻更新节点状态和孤岛、通信对数量，按题目要求模拟即可。

PS：由于本题数据规模大，务必使用快读读入数据，否则会 TLE。

- 时间复杂度：$O(m(Tlogn))$

4.2.2 C++ 代码实现

202203-4 通讯系统管理

```cpp
#include <iostream>
#include <algorithm>
#include <cstring>
#include <vector>
#include <set>
#include <map>
using namespace std;
const int maxn = 1e5 + 10;

struct Node
{
    Node(int ee, long long v) : e(ee), val(v) {}

    int e;
    long long val;

    bool operator<(const Node &n) const
    {
        if (this->val == n.val)
            return this->e < n.e;
        return this->val > n.val;
    }
};

struct Info
{
    Info(int uu, int vv, long long v) : u(uu), v(vv), val(v) {}
    int u, v;
    long long val;
};

set<Node> node[maxn];
map<pair<int, int>, long long> mem;
vector<Info> t[maxn];
int n, m, u, v, x, y, k, l, id, p, q, ln = 0, pn = 0;

inline int read()
{
    int f = 1, k = 0;
    char c = getchar();
```

```
41
42      while (c < '0' || c > '9')
43      {
44          c = getchar();
45      }
46
47      while (c >= '0' && c <= '9')
48      {
49          k = k * 10 + c - '0';
50          c = getchar();
51      }
52      return f * k;
53  }
54
55  inline bool islonely(int s)
56  {
57      if (node[s].begin()->val == 0 || node[s].begin()->e == s)
58          return true;
59      return false;
60  }
61
62  inline bool ispair(int s)
63  {
64      if (islonely(s))
65          return false;
66      if (!islonely(node[s].begin()->e) && node[node[s].begin()->e].begin()->e == s)
67          return true;
68      return false;
69  }
70
71  inline void oprt(int u, int v, long long val)
72  {
73      long long pre_val = mem[make_pair(u, v)];
74      mem[make_pair(u, v)] += val;
75
76      ln -= islonely(u);
77      pn -= ispair(u);
78
79      Node temp(v, pre_val);
80      node[u].erase(temp);
81      node[u].emplace(v, mem[make_pair(u, v)]);
82
```

```
83      ln += islonely(u);
84      pn += ispair(u);
85  }
86
87  int main()
88  {
89      n = read(), m = read();
90      ln = n;
91
92      for (int i = 1; i <= m; i++)
93      {
94          for (auto &j : t[i])
95          {
96              oprt(j.u, j.v, j.val);
97              oprt(j.v, j.u, j.val);
98          }
99
100         k = read();
101         for (int j = 1; j <= k; j++)
102         {
103             u = read(), v = read(), x = read(), y = read();
104             if (i + y <= m)
105                 t[i + y].emplace_back(u, v, -x);
106             oprt(u, v, x);
107             oprt(v, u, x);
108         }
109
110         l = read();
111         for (int j = 1; j <= l; j++)
112         {
113             id = read();
114             if (islonely(id))
115                 printf("0\n");
116             else
117                 printf("%d\n", node[id].begin()->e);
118         }
119
120         p = read(), q = read();
121         if (p)
122             printf("%d\n", ln);
123         if (q)
124             printf("%d\n", pn);
```

```
125     }
126
127     // system("pause");
128
129     return 0;
130 }
```

5 202203-5 博弈论与石子合并

- 时间限制：1.0 s
- 空间限制：512.0 MB

5.1 题目

5.1.1 问题描述

小 z 和小 c 准备玩 Nim 游戏，他们准备了一共 n 堆石子，从左到右第 i 堆石子的数量为 a_i。游戏规则如下：

两人轮流操作，每次操作可以选择将两堆相邻的石子合并，或扔掉最左边或最右边的石子，且不能不操作，直至只剩下一堆石子。

现假设小 z 希望最后剩下的石子尽量多，小 c 希望尽量少，且二人的大脑与量子计算机平分秋色，问最后这堆石子的数量。

5.1.2 子任务

子任务 1 (20 分)：$n \leq 20$，无特殊性质;
子任务 2 (20 分)：$n \leq 10^5$，每堆石子大小相等；
子任务 3 (20 分)：$n \leq 10^5$，n 是偶数，且小 z 先手；
子任务 4 (20 分)：$n \leq 10^5$，n 是偶数，且小 c 先手；
子任务 5 (5 分)：$n \leq 2000$，无特殊性质；
子任务 6 (15 分)：$n \leq 10^5$，无特殊性质；
对于 100% 的数据，$1 \leq n \leq 10^5$，$0 \leq k \leq 1$，$a_i > 0$，$\sum_{i=1}^{n} a_n \leq 10^9$。

5.2 题解

5.2.1 解法一 (100 分)

根据题意，我们可以将问题分解为两部分：

- 小 z 操作时剩余堆数为偶数：从小 c 的角度看，如果小 z 选择合并的不是中间的两堆石子，则无论之后小 z 如何操作，该堆石子都会被小 c 扔掉，所以小 z 只能合并中间两堆石子，这种情况下小 c 束手无策，只能气急败坏地尽可能多丢掉一些石子。由于小 c 只能操作 $\frac{n}{2}-1$ 次。所以问题转化为求 $\sum_{i=j}^{j+\frac{n}{2}} a_i$ $(1 \le j \le \frac{n}{2})$ 的最小值。
- 小 z 操作时剩余堆数为奇数：设 $sum_i = \sum_{i=j}^{k} a_i$ $(1 \le j \le k \le n)$，由上论述可知，此情况下无论小 z 选择合并哪两堆，小 c 都可以将其扔掉。而小 c 只能操作 $\frac{n}{2}$ 次，也就是只能扔掉 $\frac{n}{2}$ 堆石子。因此，小 z 只能广布局，将某些堆合并，同时，为了保证合并后的堆不全被小 c 扔掉，合并后的堆的数量应大于 $\frac{n}{2}$，且最后剩下的一定是数量最小的一堆。所以，问题转化成寻求一个尽可能大的 x，使 $sum_i \ge x$，同时这样的堆数应大于 $\frac{n}{2}$。

PS：对于情况二，我们很容易得出 x 即为最后的答案：

- 若 $\exists sum_i = x$，由于小 c 希望尽可能小，所以会将大于 x 的堆扔掉。
- 若 $\forall sum_i > x$，说明 $x+1$ 对于情况二也成立，此时 x 并非最终解。
- 时间复杂度：$O(nlogn)$
- 空间复杂度：$O(2n)$

5.2.2 C++ 代码实现

202203-5 博弈论与石子合并

```cpp
#include <iostream>
#include <algorithm>
#include <cstring>
using namespace std;
const int maxn = 1e5 + 10;
```

```
6
7   int n, k, a[maxn], sum[maxn], ans = 1e9 + 10;
8
9   bool judge(int x)
10  {
11      int summ = 0, cnt = 0;
12      for (int i = 1; i <= n; i++)
13      {
14          summ += a[i];
15          if (summ >= x)
16          {
17              summ = 0;
18              cnt++;
19          }
20      }
21
22      return cnt > n / 2;
23  }
24
25  int main()
26  {
27      cin >> n >> k;
28
29      for (int i = 1; i <= n; i++)
30          cin >> a[i];
31
32      if ((k && !(n % 2)) || (!k && (n % 2)))
33      {
34          for (int i = 1; i <= n; i++)
35              sum[i] = sum[i - 1] + a[i];
36
37          for (int i = 1; i <= n / 2; i++)
38          {
39              int temp = sum[i + n / 2] - sum[i - 1];
40              ans = min(ans, temp);
41          }
42      }
43      else
44      {
45          int l = 1, r = ans;
46          while (l <= r)
47          {
```

```
48              int mid = (l + r) / 2;
49              if (judge(mid))
50                  l = mid + 1;
51              else
52                  r = mid - 1;
53          }
54
55          ans = l - 1;
56      }
57
58      cout << ans << endl;
59
60      // system("pause");
61
62      return 0;
63  }
```