期末大作业 实验报告

学号：2022212408 姓名：胡宇杭

2023 年 12 月 20 日

提交结果

提交编号	用户名	姓名	试题名称	提交时间	代码长度	编程语言	评测结果	得分	时间使用	空间使用
3758739	胡宇杭	胡宇杭	星际网络II	06-26 01:43	5.166KB	CPP14	正确	100	328ms	51.10MB
3758738	胡宇杭	胡宇杭	食材运输	06-26 01:43	2.377KB	CPP14	正确	100	15ms	3.023MB

#	When	Who	Problem	Lang	Verdict	Time	Memory
211045063	Jun/25/2023 23:13 UTC+8	Lieserl	1784D - Wooden Spoon	GNU C++17	Accepted	3930 ms	394500 KB

图 1: 提交结果

目录

1 202012-4 食材运输

- 时间限制：1.0 s
- 空间限制：512.0 MB

1.1 题目

1.1.1 题目背景

在 T 市有很多个酒店，这些酒店对于不同种类的食材有不同的需求情况，莱莱公司负责每天给这些酒店运输食材。由于酒店众多，如何规划运输路线成为了一个非常重要的问题。你作为莱莱公司的顾问，请帮他们解决这个棘手的问题。

1.1.2 问题描述

T 市有 N 个酒店，由 $N-1$ 条无向边连接，构成一棵树，通过某一条边的时间取决于该边的权值。T 市有 K 种食材，每个酒店对食材的需求不同。现有 K 辆车，每辆车负责配送一种食材，且不存在重复。

每辆车都可以从不超过 M $(1 \leq M \leq K)$ 个检查点中选择一个作为出发点，现需要确定如何设置检查点和车辆的出发点，使得所有酒店等待时间最大值的最小值。

1.1.3 子任务

30% 的测试数据满足 $N \leq 10^2$，$M = K$，$K \leq 10$，保证输入数据是一条链，且节点编号满足 $u + 1 = v$。

40% 的测试数据满足 $N \leq 10^2$，$M = K$，$K \leq 10$，无特殊性质。

30% 的测试数据满足 $N \leq 10^2$，$M \leq K$，$K \leq 10$，无特殊性质。

1.2 题解

1.2.1 解法一 (65 分)

注意到 70% 的数据满足 $M = K$，等价于对每辆车都可以设置一个检查点，此时问题转化为求解出第 i 种食材在以点 u 为出发点的运输距离 dis_{i_u}，取其最

小值 (最短路径)，再对整体取最大值，即：

$$ans = max\{min(dis_{i_u}, u \in N), i \in K\}$$

而求最短路径可以通过树状 dp 实现。不难发现，除了离出发点最远的节点，其他路径都需要经过两次，因此可以通过 dp 求出经过路径的总长度，同时 DFS 出最长路径。即：$dis_{i_u} = dp[u] - max\_dis$。

该解法理论得分 70 分。(~~剩下 5 分我也不知道哪扣的~~)

1.2.2 C++ 代码实现

202012-4 食材运输 65 分代码

```cpp
#include <iostream>
#include <algorithm>
#include <cstring>
#include <ctime>
#include <cstdio>
#include <vector>
#define ll long long
using namespace std;

const int maxn = 1e2 + 10;

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

    while (c < '0' || c > '9')
        c = getchar();

    while (c >= '0' && c <= '9')
    {
        k = k * 10 + c - '0';
        c = getchar();
    }

    return f * k;
}
```

```
28
29  // Write your code below :)
30
31  struct Edge
32  {
33      Edge(int t, int v) : to(t), val(v) {}
34      int to, val;
35  };
36
37  vector<Edge> s[maxn];
38  bool req[maxn][20], flag[maxn];
39  ll N, M, K, ans = 0, dp[maxn];
40
41  inline ll dfs(int u, int fa)
42  {
43      ll maxx = 0;
44      for (auto &v : s[u])
45      {
46          if (v.to == fa)
47              continue;
48          ll temp = dfs(v.to, u);
49
50          if (flag[v.to])
51          {
52              flag[u] = true;
53              dp[u] += dp[v.to] + v.val * 2;
54              maxx = max(maxx, temp + v.val);
55          }
56      }
57
58      return maxx;
59  }
60
61  // end here
62
63  int main()
64  {
65      ios::sync_with_stdio(false);
66      cin.tie(0);
67      cout.tie(0);
68
69      // Write your code below :)
```

```
70
71      cin >> N >> M >> K;
72      int u, v, w;
73
74      for (int i = 1; i <= N; i++)
75          for (int j = 1; j <= K; j++)
76              cin >> req[i][j];
77
78      for (int i = 1; i < N; i++)
79      {
80          cin >> u >> v >> w;
81          s[u].emplace_back(v, w);
82          s[v].emplace_back(u, w);
83      }
84
85      for (int i = 1; i <= K; i++)
86      {
87          ll temp = 1e19 + 10;
88          for (int j = 1; j <= N; j++)
89          {
90              memset(dp, 0, sizeof(dp));
91              memset(flag, false, sizeof(flag));
92
93              for (int k = 1; k <= N; k++)
94                  flag[k] = req[k][i];
95
96              ll dis = dfs(j, 0);
97              temp = min(dp[j] - dis, temp);
98          }
99          ans = max(ans, temp);
100     }
101
102     cout << ans << endl;
103
104     // end here
105
106     return 0;
107 }
```

1.2.3 解法二 (100 分)

从状压 dp 的角度考虑，设 S_j 为二进制状态为 j 的食材种类的集合。则 $dp(i, S_j)$ 代表选择 i 个检查点，运送食材种类为 S_j 时的最短路径。状态转移方程为：

$$dp(i, S_j) = min\{max(dp(1, T), dp(i-1, S_j - T)), T \subseteq S_j\}$$

证明：假设已知 $dp(1, T)$, $dp(i-1, S_j-T)$, 则 $dp(i, S_j)$ 即是在 $dp(i-1, S_j-T)$ 的基础上再选一个节点去运送剩余食材种类 T。由于题目只要求我们求出等待时间最大值的最小值，所以我们只要取两者的最大值即为该状态下最大等待时间，再遍历 S_j, 选出使最大等待时间最小的 T，即为最短路径。

初始化：同解法一，用树状 dp 计算出 $dp(1, S_j)$，即：

$$dp(1, S_j) = min\{max(dis_{u_i}), i \in S_j\}, u \in N$$

- 时间复杂度：DFS 需要遍历每个节点，时间复杂度为 $O(N)$，状压 dp 最外层遍历 M 个检查点，内层循环遍历全部状态和其全部子集，时间复杂度为 $O(M * (2^K)^2)$，总时间复杂度为 $O(M * (2^K)^2)$；
- 空间复杂度：$O(M * 2^K + NK + N)$。

1.2.4 C++ 代码实现

202012-4 食材运输 100 分代码

```cpp
#include <iostream>
#include <algorithm>
#include <cstring>
#include <ctime>
#include <cstdio>
#include <vector>
#define ll long long
using namespace std;

const int maxn = 1e2 + 10;

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

```
15      char c = getchar();
16
17      while (c < '0' || c > '9')
18          c = getchar();
19
20      while (c >= '0' && c <= '9')
21      {
22          k = k * 10 + c - '0';
23          c = getchar();
24      }
25
26      return f * k;
27  }
28
29  // Write your code below :)
30
31  struct Edge
32  {
33      Edge(int t, int v) : to(t), val(v) {}
34      int to, val;
35  };
36
37  vector<Edge> s[maxn];
38  bool req[maxn][20], flag[maxn];
39  ll N, M, K, dp[maxn], f[maxn][1 << 12], initial[maxn][20];
40
41  inline ll dfs(int u, int fa)
42  {
43      ll maxx = 0;
44      for (auto &v : s[u])
45      {
46          if (v.to == fa)
47              continue;
48          ll temp = dfs(v.to, u);
49
50          if (flag[v.to])
51          {
52              flag[u] = true;
53              dp[u] += dp[v.to] + v.val * 2;
54              maxx = max(maxx, temp + v.val);
55          }
56      }
```

```
57
58      return maxx;
59  }
60
61  // end here
62
63  int main()
64  {
65      ios::sync_with_stdio(false);
66      cin.tie(0);
67      cout.tie(0);
68
69      // Write your code below :)
70
71      cin >> N >> M >> K;
72      int u, v, w;
73
74      for (int i = 1; i <= N; i++)
75          for (int j = 1; j <= K; j++)
76              cin >> req[i][j];
77
78      for (int i = 1; i < N; i++)
79      {
80          cin >> u >> v >> w;
81          s[u].emplace_back(v, w);
82          s[v].emplace_back(u, w);
83      }
84
85      for (int i = 1; i <= K; i++)
86          for (int j = 1; j <= N; j++)
87          {
88              memset(dp, 0, sizeof(dp));
89              memset(flag, false, sizeof(flag));
90
91              for (int k = 1; k <= N; k++)
92                  flag[k] = req[k][i];
93
94              ll dis = dfs(j, 0);
95              initial[j][i] = dp[j] - dis;
96          }
97
98      for (int i = 1; i < (1 << K); i++)
```

```
    {
        ll minn = 1e19 + 10;
        for (int j = 1; j <= N; j++)
        {
            ll temp = 0;
            for (int k = 1; k <= K; k++)
            {
                if (i & (1 << (k - 1)))
                    temp = max(temp, initial[j][k]);
            }
            minn = min(minn, temp);
        }
        f[1][i] = minn;
    }

    for (int i = 2; i <= M; i++)
        for (int j = 1; j < (1 << K); j++)
        {
            f[i][j] = 1e19 + 10;
            for (int k = j; k; k = (k - 1) & j)
                f[i][j] = min(f[i][j], max(f[1][k], f[i - 1][j ^ k]));
        }

    cout << f[M][(1 << K) - 1] << endl;

    // end here

    return 0;
}
```

2 202303-4 星际网络 II

- 时间限制：2.0 s
- 空间限制：1.0 GB

2.1 题目

2.1.1 问题描述

管理一系列由 n 位二进制位组成 ($n \ mod \ 16 = 0$)，用 $IPv6$ 方法表示的地址。记 $num(s)$ 为地址 s 按高位在前、低位在后组成的 n 位二进制数，称一段连续区间的一系列地址。现需要支持对地址进行以下操作。

- 1 id l r：用户 id 申请地址在 l r 的一段连续地址，当且仅当该段地址未被分配或该段地址只被部分 (不完全) 分配给用户 id 时能申请成功。
- 2 s：检查地址 s 被分配给了哪个用户。
- 3 l r：检查 l r 范围内的所有地址是否全被分配给了某个用户。

2.1.2 子任务

数据范围

对于所有数据，$n \le 512$, $q \le 5 \times 10^4$, n 为 16 的倍数，$id \le q$，对于操作 1,3 保证 $num(l) \le num(r)$。

测试点编号	$n \le$	$q \le$	特殊性质
1 ~ 4	16	200	无
5 ~ 6	64	200	无
7 ~ 9	512	200	无
10 ~ 11	16	20000	无
12 ~ 13	64	50000	无
14 ~ 16	512	50000	所有操作 1 的 id 互不相同
17 ~ 20	512	50000	无

图 2: 子任务

2.2 题解

2.2.1 解法一 (100 分)

题目要求涉及到区间修改、单点查询、区间查询，可以考虑使用线段树进行维护。节点储存当前区间的左右端点 l r，已被占用的区间长度 cnt 以及该区间 id 的最大最小值。对于每次操作：

- 操作 1：若该段区间 $cnt = 0$，或 $cnt\ != r - l + 1$ 且 $id_{min} = id_{max} = id_{now}$，则该区间可修改。
- 操作 2：二分搜索查找即可。
- 操作 3：若该段区间 $cnt = r - l + 1$ 且 $id_{min} = id_{max}$，则全被分配给了某用户。

同时，我们注意到：子任务中存在 $n = 512$ 的数据，而 $long\ long$ 的范围只到 $2^{64} - 1$，并且 2^{512} 量级肯定会爆内存。但子任务中总操作次数 $q \leq 50000$，因此可以使用离散化处理 (~~和 202112-4 一模一样，简直白给~~)。

由于题目很贴心的使用了 $IPv6$ 的地址表示方法，所以可以直接比较地址字符串来确定相对大小，处理下进位即可。最后，需要注意要把 $l + 1$、$s + 1$ 一并离散化，否则会出现诸如 $[1, 2], [4, 5]$ 在离散化后两区间相邻的问题。

- 时间复杂度：设离散化后共有 n 个点，线段树单次查询，修改的时间复杂度均为 $O(logn)$，共有 q 次查询，总时间复杂度为 $O(q * logn)$；
- 空间复杂度：$O(4n)$。

2.2.2 C++ 代码实现

202303-4 星际网络 II

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

```cpp
#define ll long long
using namespace std;

const int maxn = 8e5 + 10;
const int INF = 1e9 + 10;

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

    while (c < '0' || c > '9')
        c = getchar();

    while (c >= '0' && c <= '9')
    {
        k = k * 10 + c - '0';
        c = getchar();
    }

    return f * k;
}

// Write your code below :)

#define ls u << 1
#define rs u << 1 | 1

struct Node
{
    int l, r, minn, maxx, cnt, laz;
} t[maxn << 2];

struct Oprt
{
    Oprt(int tt, int idd, string L, string R, string S) : type(tt), id(idd), l(L), r(R), s(S
        ) {}
    int type, id, lr, rr, sr;
    string l, r, s;
};

vector<Oprt> oprt;
```

```
48 vector<string> cord;
49 int n, q, m;
50
51 inline void pushup(int u)
52 {
53     t[u].cnt = t[ls].cnt + t[rs].cnt;
54     t[u].maxx = max(t[ls].maxx, t[rs].maxx);
55     t[u].minn = min(t[ls].minn, t[rs].minn);
56 }
57
58 inline void pushdown(int u)
59 {
60     if (t[u].l == t[u].r)
61         return;
62     if (t[u].laz)
63     {
64         t[ls].laz = t[rs].laz = t[u].laz;
65         t[ls].cnt = t[ls].r - t[ls].l + 1;
66         t[rs].cnt = t[rs].r - t[rs].l + 1;
67         t[ls].maxx = t[ls].minn = t[u].laz;
68         t[rs].maxx = t[rs].minn = t[u].laz;
69         t[u].laz = 0;
70     }
71 }
72
73 inline void build(int u, int l, int r)
74 {
75     t[u] = {l, r, INF, -INF, 0, 0};
76     if (l == r)
77         return;
78
79     int mid = l + r >> 1;
80
81     build(ls, l, mid);
82     build(rs, mid + 1, r);
83
84     pushup(u);
85 }
86
87 inline void query(int u, int l, int r, Node &res)
88 {
89     if (t[u].l == l && t[u].r == r)
```

```
90      {
91          res.cnt += t[u].cnt;
92          res.minn = min(res.minn, t[u].minn);
93          res.maxx = max(res.maxx, t[u].maxx);
94          return;
95      }
96
97      pushdown(u);
98
99      int mid = t[u].l + t[u].r >> 1;
100
101     if (l <= mid)
102         query(ls, l, min(mid, r), res);
103     if (r > mid)
104         query(rs, max(mid + 1, l), r, res);
105     return;
106 }
107
108 inline int query_2(int u, int s)
109 {
110     if (t[u].l == t[u].r)
111         return t[u].minn;
112
113     pushdown(u);
114
115     int mid = t[u].l + t[u].r >> 1;
116
117     if (s <= mid)
118         return query_2(ls, s);
119     else
120         return query_2(rs, s);
121 }
122
123 inline void update(int u, int l, int r, int id)
124 {
125     if (t[u].l == l && t[u].r == r)
126     {
127         t[u].minn = t[u].maxx = id;
128         t[u].cnt = t[u].r - t[u].l + 1;
129         t[u].laz = id;
130         return;
131     }
```

```
132
133     pushdown(u);
134
135     int mid = t[u].l + t[u].r >> 1;
136
137     if (l <= mid)
138         update(ls, l, min(mid, r), id);
139     if (r > mid)
140         update(rs, max(mid + 1, l), r, id);
141
142     pushup(u);
143 }
144
145 char carry(char ch)
146 {
147     if (ch == '9')
148         return 'a';
149     else if (ch == 'f')
150         return '0';
151     else
152         return ch + 1;
153 }
154
155 string add(string s)
156 {
157     for (int i = s.size() - 1; i >= 0; i--)
158     {
159         if (s[i] == ':')
160             continue;
161         s[i] = carry(s[i]);
162         if (s[i] != '0')
163             break;
164     }
165     return s;
166 }
167
168 void discretization()
169 {
170     sort(cord.begin(), cord.end());
171     m = unique(cord.begin(), cord.end()) - cord.begin();
172     cord.resize(m);
173
```

```
174     for (auto &op : oprt)
175     {
176         if (op.type == 1 || op.type == 3)
177         {
178             op.lr = lower_bound(cord.begin(), cord.end(), op.l) - cord.begin() + 1;
179             op.rr = lower_bound(cord.begin(), cord.end(), op.r) - cord.begin() + 1;
180         }
181         else
182             op.sr = lower_bound(cord.begin(), cord.end(), op.s) - cord.begin() + 1;
183     }
184 
185     return;
186 }
187 
188 // end here
189 
190 int main()
191 {
192     ios::sync_with_stdio(false);
193     cin.tie(0);
194     cout.tie(0);
195 
196     // Write your code below :)
197 
198     cin >> n >> q;
199     int type, id;
200     string l, r, s;
201 
202     for (int i = 1; i <= q; i++)
203     {
204         cin >> type;
205 
206         if (type == 1)
207         {
208             cin >> id >> l >> r;
209             oprt.emplace_back(type, id, l, r, " ");
210         }
211         else if (type == 2)
212         {
213             cin >> s;
214             oprt.emplace_back(type, id, " ", " ", s);
215         }
```

```
216         else
217         {
218             cin >> l >> r;
219             oprt.emplace_back(type, 0, l, r, " ");
220         }
221         if (type == 1 || type == 3)
222             cord.push_back(l), cord.push_back(r), cord.push_back(add(r));
223         else
224             cord.push_back(s), cord.push_back(add(s));
225     }
226 
227     discretization();
228     build(1, 1, m);
229 
230     for (auto &op : oprt)
231     {
232         if (op.type == 1)
233         {
234             Node res = {op.lr, op.rr, INF, -INF, 0, 0};
235             query(1, op.lr, op.rr, res);
236             if (res.cnt == 0 || res.maxx == res.minn && res.minn == op.id && res.cnt != op.
                    rr - op.lr + 1)
237             {
238                 cout << "YES" << endl;
239                 update(1, op.lr, op.rr, op.id);
240             }
241             else
242                 cout << "NO" << endl;
243         }
244         else if (op.type == 2)
245         {
246             int temp = query_2(1, op.sr);
247             if (temp == INF)
248                 cout << 0 << endl;
249             else
250                 cout << temp << endl;
251         }
252         else
253         {
254             Node res = {op.lr, op.rr, INF, -INF, 0, 0};
255             query(1, op.lr, op.rr, res);
256             if (res.cnt == op.rr - op.lr + 1 && res.maxx == res.minn)
```

```
257                 cout << res.minn << endl;
258             else
259                 cout << 0 << endl;
260         }
261     }
262 
263     // end here
264 
265     return 0;
266 }
```

3 Codeforces 1786F Wooden Spoon

- 时间限制：4.0 s
- 空间限制：512.0 MB

3.1 题目

3.1.1 问题描述

有 2^n 个人 (编号 $1 \sim 2^n$) 参加比赛，比赛形式类似一颗完全二叉树，并且每次比赛总是二者中编号较小的一方赢得胜利。同时，当参赛者满足以下条件时会获得安慰奖 *Wooden Spoon*。

- 该参赛者 A 输了他的第一场比赛；
- 打败上一位参赛者 A 的参赛者 B 输了他的第二场比赛；
- 打败上一位参赛者 B 的参赛者 C 输了他的第三场比赛；
- ……；
- 打败上一位参赛者的参赛者输了最终决赛。

试确定对于每一位参赛者，其能获得 *Wooden spoon* 时的比赛安排方式的种类 (*modulo* 998244353)。

3.1.2 子任务

一共有 20 个测试点，对于第 i 个测试点，有 $n = i$。

3.2 题解

3.2.1 解法一

根据题意，我们不难发现：

- 从根节点到获得 *Wooden Spoon* 叶子节点 (假定权值为 i) 的路径满足节点权值单调递增的性质，即：$1 = x_1 < x_2 < \ldots < x_n < i$。

- 某一子树的根节点的权值一定是该子树中权值最小的叶子节点。

对于上述路径，考虑使用 dp 求解。设 $dp(i,j)$ 表示路径 $x_1 -> x_i = j$ 时的方案数，即权值为 j 的节点在倒数第 $i-1$ 场比赛输掉的方案数。则有：

$$dp(i,j) = 2 * 2^{n-i}!\ C_{2^n-2^{n-i}-j}^{2^{n-i}-1} \sum_{k=1}^{j-1} dp(i-1,k)$$

证明：

- $2 * 2^{n-i}!$：由完全二叉树的性质，节点 j 高度为 i，则其某一子树的叶子节点共有 2^{n-i} 个；由性质二，无论这些叶子节点如何排列，最终胜出的一定为 j，所以对 j 所在子树的节点求全排列数，即 $2^{n-i}!$；同时，j 节点可能出现在左右子树，有两种可能，因此最终结果为 $2 * 2^{n-i}!$。

- $C_{2^n-2^{n-i}-j}^{2^{n-i}-1}$：目前已算出 j 所在子树的 2^{n-1} 个叶子节点的排列方式，还需确定 j 的对手的子树；由性质二：以 j 为根节点的子树的叶子节点 k 权值一定满足 $j < k \leq 2^n$，又因为 j 赢得该场比赛，所以其对手节点权值肯定大于 j，此时只需要选择剩下 $2_{n-i} - 1$ 个叶子节点，同时，注意不能选择 j 所在子树的 2_{n-i} 叶子节点。

- $\sum_{k=1}^{j-1} dp(i-1,k)$：选择一个权值小于 j 的节点作为 j 的对手，保证 j 失败，其方案数共有 $\sum_{k=1}^{j-1} dp(i-1,k)$ 种。

最终答案为：$\sum_{j=1}^{i-1} dp(n,j)$，这是因为 $dp(n,j)$ 代表沿着上述路径，但只赢了第一场比赛的方案数，而我们要求的是沿上述路径第一场也输了的方案数。此时，对于编号为 j 的参赛者，他可以选择与玩家 i $(i < j)$ 比赛来保证第一场比赛失败。

PS：由于题目要求对结果取模，所以可以用快速模指数运算和费马小定理求逆元 (~~死去的离散又开始攻击我~~)。

- 时间复杂度：$O(n * 2^n)$；

- 空间复杂度：$O((2n+2) * 2^n)$。

3.2.2 C++ 代码实现

Codeforces 1786F Wooden Spoon

```cpp
#include <iostream>
#include <algorithm>
#include <cstring>
#include <ctime>
#include <cstdio>
#define ll long long
using namespace std;

const int maxn = 1050000;
const int mod = 998244353;

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

    while (c < '0' || c > '9')
        c = getchar();

    while (c >= '0' && c <= '9')
    {
        k = k * 10 + c - '0';
        c = getchar();
    }

    return f * k;
}

// Write your code below :)

ll n, N, fac[maxn], ifac[maxn], dp[23][maxn], sum[23][maxn];

inline ll q_pow(ll x, ll p)
{
    ll res = 1;
    while (p)
    {
        if (p & 1)
```

```
39              res = (res * x) % mod;
40          x = x * x % mod;
41          p = p >> 1;
42      }
43
44      return res;
45  }
46
47  inline ll c(ll m, ll n)
48  {
49      if (m < 0 || n < 0 || m > n)
50          return 0;
51      return fac[n] * ifac[m] % mod * ifac[n - m] % mod;
52  }
53
54  // end here
55
56  int main()
57  {
58      ios::sync_with_stdio(false);
59      cin.tie(0);
60      cout.tie(0);
61
62      // Write your code below :)
63
64      cin >> n;
65
66      N = 1 << n;
67      dp[0][0] = 1, fac[0] = 1;
68
69      for (int i = 1; i <= N; i++)
70          fac[i] = fac[i - 1] * i % mod;
71
72      ifac[N] = q_pow(fac[N], mod - 2);
73      for (int i = N - 1; i >= 0; i--)
74          ifac[i] = ifac[i + 1] * (i + 1) % mod;
75
76      for (ll i = 0; i <= N; i++)
77          sum[0][i] = 1;
78
79      for (ll i = 1; i <= n; i++)
80          for (ll j = 1; j <= N; j++)
```

```
81          {
82              dp[i][j] = 2 * fac[1 << (n - i)] % mod * c((1 << (n - i)) - 1, N - (1 << (n - i)
                    ) - j) % mod * sum[i - 1][j - 1] % mod;
83              sum[i][j] = (sum[i][j - 1] + dp[i][j]) % mod;
84          }
85
86      for (ll i = 1; i <= N; i++)
87          cout << sum[n][i - 1] << endl;
88
89      // end here
90
91      return 0;
92  }
```