二维01背包+完全背包

人员

于家瑞、洪晨棋、洪晨栋、于霄龙、陶汇笙 到课, 王恩泽、郭栩睿 线上

上周作业检查

上周作业链接: https://cppoj.kids123code.com/contest/1906

比赛概况 | 题目列表 | 选择题列表 | 提交记录 | ★ 实时榜单 | ★ 选择题排行榜

王向东老师周六十点半C++01背包练习

刷新

#	用户名	姓名	编程分	时间	A	B	C	D	E
1	yuxiaolong	于霄龙	500	218	100	100	100	100	100
2	hongchendong	洪晨栋	500	230	100	100	100	100	100
3	hongchenqi	洪晨棋	500	293	100	100	100	100	100
4	yujiarui	于家瑞	500	319	100	100	100	100	100
5	taohuisheng	陶汇笙	400	206	100	100	100	100	
6	guoxurui	郭栩睿	400	231	100	100	100	100	
7	cuichenhe	崔宸赫	400	277	100	100	100	100	
8	wangenze	王恩泽	100	38	100				

本周作业

https://cppoj.kids123code.com/contest/2016 (课上讲了 A ~ F 这些题, 课后作业是 G 题)

课堂表现

今天课上讲了 二维01背包 和 完全背包 这 2 个知识点, 这 2 个知识点整体都比较简单, 同学们课上都掌握的比较好

今天的 F 题会复杂一些, 课上老师讲完这个题了, 同学们都没写完, 同学们课下要好好写写这个题。

课堂内容

[USACO2.2] 集合 Subset Sums (上周作业)

题目让总和平分, 总和是 sum, 平分就是看凑出来 sum/2 时一共有多少方案

这样会有重复, 可以把方案数除以 2, 也可以考虑不用 n 这个数时求方案数是多少

```cpp
#include <bits/stdc++.h>

using namespace std;

const int maxn = 1000 + 5;
int f[maxn];
```

```
int main()
{
  int n; cin >> n;
  int sum = (1+n)*n / 2;
  if (sum&1) { cout << 0 << endl; return 0; }

  sum /= 2;
  f[0] = 1;
  for (int i = 1; i <= n-1; ++i) {
    for (int j = sum; j >= i; --j) f[j] += f[j-i];
  }
  cout << f[sum] << endl;
  return 0;
}
```

装备运输

f[i][j]: 当体积是 i, 重量是 j 时, 能得到的最大火力值是多少

```
#include <bits/stdc++.h>

using namespace std;

typedef long long LL;
const int maxn = 500 + 5;
LL f[maxn][maxn];

int main()
{
  int V, G, n; cin >> V >> G >> n;
  while (n -- ) {
    int t, v, g; cin >> t >> v >> g;
    for (int i = V; i >= v; --i) {
      for (int j = G; j >= g; --j) f[i][j] = max(f[i][j], f[i-v][j-g]+t);
    }
  }
  cout << f[V][G] << endl;
  return 0;
}
```

NASA的食物计划

f[i][j]: 当体积是 i, 质量是 j 时, 所含的卡路里最大是多少

```
#include <bits/stdc++.h>

using namespace std;
```

```
const int maxn = 400 + 5;
int f[maxn][maxn];

int main()
{
  int H, T; cin >> H >> T;
  int n; cin >> n;
  while (n -- ) {
    int h, t, k; cin >> h >> t >> k;
    for (int i = H; i >= h; --i) {
      for (int j = T; j >= t; --j) f[i][j] = max(f[i][j], f[i-h][j-t]+k);
    }
  }
  cout << f[H][T] << endl;
  return 0;
}
```

疯狂的采药

完全背包模板, 每个物品可以无限选, 所以把循环改为从前往后遍历即可

```
#include <bits/stdc++.h>

using namespace std;

typedef long long LL;
const int maxn = 1e7 + 5;
LL f[maxn];

int main()
{
  int T, n; cin >> T >> n;
  while (n -- ) {
    int a, b; cin >> a >> b;
    for (int i = a; i <= T; ++i) f[i] = max(f[i], f[i-a]+b);
  }
  cout << f[T] << endl;
  return 0;
}
```

A+B Problem（再升级）

f[i]: 正好凑成 i 这个数时, 有多少方案

每个素数可以不选或者选任意次, 因此套用完全背包模板即可

```
#include <bits/stdc++.h>

using namespace std;
```

```
vector<int> primes;
bool isPrime(int x) {
  if (x <= 1) return false;
  for (int i = 2; i*i <= x; ++i) {
    if (x%i == 0) return false;
  }
  return true;
}

typedef long long LL;
const int maxn = 1000 + 5;
LL f[maxn];

int main()
{
  int n; cin >> n;
  for (int i = 1; i <= n; ++i) {
    if (isPrime(i)) primes.push_back(i);
  }

  f[0] = 1;
  for (int i : primes) {
    for (int j = i; j <= n; ++j) f[j] += f[j-i];
  }
  cout << f[n] << endl;
  return 0;
}
```

神奇的四次方数

f[i]: 凑出 i 这个值时, 最少用到了几个四次方数

每个四次方数可以不用或者用任意次, 套完全背包模板即可

```
#include <bits/stdc++.h>

using namespace std;

const int maxn = 1e5 + 5;
int f[maxn];

int main()
{
  int m; cin >> m;
  memset(f, 0x3f, sizeof(f)); f[0] = 0;
  for (int i = 1; ; ++i) {
    int t = i*i*i*i;
    if (t > m) break;
    for (int j = t; j <= m; ++j) f[j] = min(f[j], f[j-t]+1);
  }
```

```
  cout << f[m] << endl;
  return 0;
}
```

[蓝桥杯 2025 省 Java B] 数组翻转

找到每个数出现的最长的连续两段的长度

```
#include <bits/stdc++.h>

using namespace std;

typedef long long LL;
const int maxn = 1e6 + 5;
int w[maxn];
vector<int> vec[maxn];

int main()
{
  int n; cin >> n;
  for (int i = 1; i <= n; ++i) cin >> w[i];
  int cnt = 1;
  for (int i = 2; i <= n; ++i) {
    if (w[i] == w[i-1]) ++cnt;
    else vec[w[i-1]].push_back(cnt), cnt = 1;
  }
  vec[w[n]].push_back(cnt);

  LL res = 0;
  for (int i = 1; i < maxn; ++i) {
    sort(vec[i].begin(), vec[i].end()), reverse(vec[i].begin(), vec[i].end());
    int len = 0;
    if ((int)vec[i].size() >= 1) len += vec[i][0];
    if ((int)vec[i].size() >= 2) len += vec[i][1];
    res = max(res, (LL)len * i);
  }
  cout << res << endl;
  return 0;
}
```