Paradigmas de Resolução de Problemas

Programação Dinâmica – Mochila Binária: Exercícios Resolvidos

Prof. Edson Alves - UnB/FGA

2020

Sumário

SPOJ WACHOVIA – Wachovia Bank

Problema

Danilo Gheyi is a renowned bank robber. He is known worldwide for accomplishing the most profitable bank robbery, in Fortaleza, Ceará. Danilo and his friends dug a tunnel to get into the main chest. There were some bags, with different amounts of money or jewelry and weight. They had to leave about 50% of the total value, since the truck couldn't carry all the bags.

Danilo wasn't caught, and to show that he can do it all again, he is planning a robbery to one of the safer banks in USA – the Wachovia Bank. He wants your help to maximize the amount stolen, avoiding a huge loss as it happened in Fortaleza.

Write a program that, given the maximum weight the truck is able to carry and the information about each bag in the bank, determine the maximum value that Danilo can steal.

Entrada e saída

Input

The input consists of several instances. There is an integer N ($1 \leq N \leq 200$) in the first line; it stands for the number of instances. The first line of each instance contains two integers, K and M ($8 \leq K \leq 1000$ and $1 \leq M \leq 50$) representing, respectively, the maximum weight the truck can handle and the amount of bags in the bank. The next M lines describe each bag with two integers A and B ($8 \leq A \leq 200$ and $1 \leq B \leq 25$): the weight and the value of the bag, respectively.

Output

For each instance output a sentence "`Hey stupid robber, you can get` P.", and P represents the maximum value Danilo can steal.

Exemplo de entradas e saídas

Sample Input

```
3
34 5
178 12
30 1
13 7
34 8
87 6
900 1
900 25
100 10
27 16
131 9
132 17
6 5
6 23
56 21
100 25
1 25
25 25
100 2
```

Sample Output

```
Hey stupid robber, you can get 8.
Hey stupid robber, you can get 25.
Hey stupid robber, you can get 99.
```

Solução $O(KM)$

- O problema apresentado pode ser modelado como um problema da mochila binária
- O caminhão é a mochila do problema, com capacidade K
- Os elementos da mochila binária são os pertences a serem subtraídos do banco
- Os pesos e os valores dos elementos são A e B, respectivamente
- Estabelecida estas correspondências, basta aplicar a solução de programação dinâmica para o problema da mochila, sem alteração alguma
- Portanto, a complexidade da solução será $O(KM)$

Solução $O(KM)$

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

using namespace std;
using ii = pair<int, int>;
using ll = long long;

const int MAXN { 60 }, MAXM { 1010 };

ll st[MAXN][MAXM];

ll dp(int i, int m, int M, const vector<ii>& cs)
{
    if (i < 0)
        return 0;

    if (st[i][m] != -1)
        return st[i][m];

    auto res = dp(i - 1, m, M, cs);
    auto w = cs[i].first, v = cs[i].second;

```

Solução $O(KM)$

```
22      if (w <= m)
23          res = max(res, dp(i - 1, m - w, M, cs) + v);
24
25      st[i][m] = res;
26      return res;
27  }
28
29  ll knapsack(int M, const vector<ii>& cs)
30  {
31      memset(st, -1, sizeof st);
32
33      return dp((int) cs.size() - 1, M, M, cs);
34  }
35
36  int main()
37  {
38      ios::sync_with_stdio(false);
39
40      int T;
41      cin >> T;
42
```

Solução $O(KM)$

```
43      while (T--)
44      {
45          int M, N;
46          cin >> M >> N;
47
48          vector<ii> cs(N);
49
50          for (int i = 0; i < N; ++i)
51              cin >> cs[i].first >> cs[i].second;
52
53          cout << "Hey stupid robber, you can get "
54               << knapsack(M, cs) << ".\n";
55      }
56
57      return 0;
58  }
```

OJ 990 – Diving for Gold

Problema

John is a diver and a treasure hunter. He has just found the location of a pirate ship full of treasures. The sofisticated sonar system on board his ship allows him to identify the location, depth and quantity of gold in each suken treasure. Unfortunatelly, John forgot to bring a GPS device and the chances of ever finding this location again are very slim so he has to grab the gold now. To make the situation worse, John has only one compressed air bottle.

John wants to dive with the compressed air bottle to recover as much gold as possible from the wreck. Write a program John can use to select which treasures he should pick to maximize the quantity of gold recovered.

Problema

The problem has the following restrictions:

- There are n treasures $\{(d_1, v_1), (d_2, v_2), \ldots, (d_n, v_n)\}$ represented by the pair (depth, quantity of gold). There are at most 30 treasures.
- The air bottle only allows for t seconds under water. t is at most 1000 seconds.
- In each dive, John can bring the maximum of one treasure at a time.
- The descent time is $td_i = w \times d_i$, where w is an integer constant.
- The ascent time is $ta_i = 2w \times d_i$, where w is an integer constant.
- Due to instrument limitations, all parameters are integer.

Entrada e saída

Input

The input to this program consists of a sequence of integer values. Input contains several test cases. The first line of each dataset should contain the values t and w. The second line contains the number of treasures. Each of the following lines contains the depth d_i and quantity of gold v_i of a different treasure.

A blank line separates each test case.

Note:

In this sample input, the bottle of compressed air has a capacity of 200 seconds, the constant w has the value 4 and there are 3 treasures, the first one at a depth of 10 meters and worth 5 coins of gold, the second one at a depth of 10 meters and worth 1 coin of gold, and the third one at 7 meters and worth 2 coins of gold.

Entrada e saída

Output

The first line of the output for each dataset should contain the maximum amount of gold that John can recover from the wreck. The second line should contain the number of recovered treasures. Each of the following lines should contain the depth and amount of gold of each recovered treasure. Treasures should be presented in the same order as the input file.

Print a blank line between outputs for different datasets.

Exemplo de entradas e saídas

Sample Input

```
210 4
3
10 5
10 1
7 2
```

Sample Output

```
7
2
10 5
7 2
```

Solução $O(TN)$

- Este é um problema de mochila binária cuja solução demanda não apenas o valor ótima, mas também a lista dos elementos selecionados
- O tempo máximo de mergulho T é a capacidade da mochila
- O valor de cada elemento é v_i
- Como cada mergulho só permite reaver um único elemento, o "peso" do elemento é a soma do tempo de mergulho com o tempo de retorno à superfície
- Assim,

$$w_i = w \times d_i + 2w \times d_i = 3wd_i$$

- Para evitar o veredito WA, não se esqueça de incluir uma linha em branco entre as saídas de dois casos de teste consecutivos

Solução $O(TN)$

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

using namespace std;
using ii = pair<int, int>;

pair<int, vector<int>> solve(int T, int W, int N, vector<ii>& cs)
{
    vector<vector<int>> st(N + 1, vector<int>(T + 1, 0));
    vector<vector<int>> ps(N + 1, vector<int>(T + 1, -1));

    for (int i = 1; i <= N; ++i)
        cs[i].first *= 3*W;

    for (int i = 1; i <= N; ++i)
    {
        for (int t = 1; t <= T; ++t)
        {
            st[i][t] = st[i - 1][t];
            ps[i][t] = 0;

            auto [d, v] = cs[i];
```

Solução $O(TN)$

```
22
23              if (d <= t and st[i - 1][t - d] + v > st[i][t])
24              {
25                  st[i][t] = st[i - 1][t - d] + v;
26                  ps[i][t] = 1;
27              }
28          }
29      }
30
31      int t = T;
32      vector<int> is;
33
34      for (int i = N; i >= 1; --i)
35      {
36          if (ps[i][t])
37          {
38              is.push_back(i);
39              t -= cs[i].first;
40          }
41      }
42
```

Solução $O(TN)$

```
43      reverse(is.begin(), is.end());
44
45      return { st[N][T], is };
46  }
47
48  int main()
49  {
50      ios::sync_with_stdio(false);
51
52      int T, W, first = 1;
53
54      while (cin >> T >> W)
55      {
56          int N;
57          cin >> N;
58
59          vector<ii> cs(N + 1);
60
61          for (int i = 1; i <= N; ++i)
62              cin >> cs[i].first >> cs[i].second;
63
```

Solução $O(TN)$

```
64          auto [ans, is] = solve(T, W, N, cs);
65
66          if (not first)
67              cout << '\n';
68
69          cout << ans << '\n';
70          cout << is.size() << '\n';
71
72          for (auto i : is)
73              cout << cs[i].first / (3*W) << ' ' << cs[i].second << '\n';
74
75          first = 0;
76      }
77
78      return 0;
79  }
```

Referências

1. SPOJ WACHOVIA – Wachovia Bank
2. OJ 990 – Diving for Gold