Московский авиационный институт (национальный исследовательский университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

Лабораторная работа №7 по курсу «Дискретный анализ»

Студент: Меджидли И. И. о
Преподаватель: Михайлова С. А
Группа: М8О-201Б-21

Дата: 25.02.2024
Оценка:
Подпись:

Москва, 2024

Лабораторная работа №7

Задача: У вас есть рюкзак, вместимостью m, а так же n предметов, у каждого из которых есть вес w_i и стоимость c_i. Необходимо выбрать такое подмножество I из них, чтобы:

· $\sum_{i \in I} w_i \leq m$

· $(\sum_{i \in I} c_i) * |I|$

является максимальной из всех возможных. |I| – мощность множества I.

1 Описание

Эту задачу можно решить с помощью метода динамического программирования.

Динамическое программирование — метод решения задачи путём её разбиения на несколько одинаковых подзадач, рекуррентно связанных между собой.

Рассмотрим идею решения задачи:

Код использует динамическое программирование для нахождения оптимального решения. Он создает трехмерный массив dp, в котором dp[i][j][k] хранит максимальную суммарную стоимость k предметов из первых i, при условии, что суммарный вес не превышает j. Также он создает трехмерный массив path, в котором path[i][j][k] хранит координаты предыдущей ячейки, из которой было получено значение dp[i][j][k].

Для заполнения массива dp, код перебирает все возможные значения i, j и k, и сравнивает два варианта: включить i-й предмет в рюкзак или не включать. Если вес i-го предмета больше, чем j, то он не может быть включен, и dp[i][j][k] = dp[i-1][j][k]. Если же вес i-го предмета не больше, чем j, то нужно выбрать максимум из двух вариантов: dp[i-1][j][k] (не включать i-й предмет) и (dp[i-1][j-arr[i].first][k-1] / (k-1) + arr[i].second) * k (включить i-й предмет и увеличить среднюю стоимость на его стоимость). В массиве path записывается соответствующая предыдущая ячейка для каждого выбора.

После заполнения массива dp, код находит максимальное значение, двигаясь по всем возможным значениям j, k dp[n][j][k], в нем и запоминает его координаты. Затем он восстанавливает оптимальное подмножество предметов, следуя по массиву path от максимальной ячейки к начальной. Он добавляет номера предметов в вектор res, если они были включены в рюкзак.

В конце код выводит максимальную суммарную стоимость и номера предметов в рюкзаке.

2 Исходный код

```
#include <iostream>
#include <vector>

using namespace std;

int main()
{ ios::sync_with_stdio(false); cin.tie(0);
    int n, m, c, M;
    unsigned long long value = 0;
    int maxf, maxs, maxth;
    cin >> n >> M;
    vector <pair <int, int>> arr(n+1);
    for(int i = 1; i <= n; ++i){
        cin >> m >> c;
        arr[i] = make_pair(m, c);
    }
    vector<vector<vector<unsigned long long>>> dp(n+1, vector<vector<unsigned long long
        >>(M+1, vector<unsigned long long>(n+1)));
    vector<vector<vector<pair<int, int>>>> path(n+1, vector<vector<pair<int, int>>>(M
        +1, vector<pair<int, int>>(n+1)));
    for(int i = 1; i <= n; ++i){
        for(int j = 1; j <= M; ++j){
            for(int k = 1; k <= i; ++k){
            if(j < arr[i].first){
                dp[i][j][k] = dp[i-1][j][k];
                path[i][j][k] = {i-1, j};
            }
            else{
                if((k == 1 ) || (dp[i-1][j-arr[i].first][k-1] == 0)){
                if(dp[i-1][j][k] > arr[i].second){
                    dp[i][j][k] = dp[i-1][j][k];
                    path[i][j][k] = {i-1, j};
                }
                else{
                    if(dp[i][j][1] < arr[i].second){
                    dp[i][j][1] = arr[i].second;
                    path[i][j][1] = {i-1, j-arr[i].first};
                    }
                }
                }
                else if(dp[i-1][j][k] > (dp[i-1][j-arr[i].first][k-1] / (k-1) + arr[i].
                    second) * k){
                    dp[i][j][k] = dp[i-1][j][k];
                    path[i][j][k] = {i-1, j};
                    }
                else{
                    dp[i][j][k] = (dp[i-1][j-arr[i].first][k-1] / (k-1) + arr[i].second)
```

```
                        * k;
45                  path[i][j][k] = {i-1, j-arr[i].first};
46                  }
47              }
48          }
49          }
50      }
51
52
53      for(int i = n; i <= n; ++i){
54          for(int j = 1; j <= M; ++j){
55              for (int k = 1; k <= n; ++k){
56                  if(value < dp[i][j][k]){
57                      value = dp[i][j][k];
58
59                      maxf = i; maxs = j; maxth = k;
60                  }
61              }
62          }
63      }
64
65
66      int i = maxf, j = maxs, k = maxth;
67      vector<int> res;
68      while((i > 0 && j > 0 && k > 0) && (dp[i][j][k] != 0)){
69          if(path[i][j][k].second == j){
70              --i;
71          }
72          else{
73          res.push_back(i);
74          int ii = i;
75          i = path[i][j][k].first;
76          j = path[ii][j][k].second;
77          k-=1;
78          }
79      }
80
81      cout << value << "\n";
82
83      for(int i = res.size() - 1; i >= 0; --i){
84          if(i == 0) {cout << res[i]; break;}
85          cout << res[i] << " ";
86      }
87
88      return 0;
89  }
```

3 Тесты

Тестировать программу буду ручным способом.

```
imedzhidli@imedzhidli:~/Desktop/DA/LABA7$ ./lab7
3 6
2 1
5 4
4 2
6
1 3
imedzhidli@imedzhidli:~/Desktop/DA/LABA7$ ./lab7
5 15
6 5
4 3
3 1
2 3
5 6
52
2 3 4 5
imedzhidli@imedzhidli:~/Desktop/DA/LABA7$
```

Как можно заметить, все работает верно.

4 Выводы

При выполнении седьмой лабораторной работы по курсу «Дискретный анализ» я познокомился с таким методом решения задач на оптимизацию, как динамическое программирование, смог решить усовершенственный вариант "Задачи о рюкзаке". Надеюсь, что полученные новые знания мне понадобятся в дальнейшем.