Московский авиационный институт (национальный исследовательский университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

Лабораторная работа №8 по курсу «Дискретный анализ»

Студент: А. О. Ларченко
Преподаватель: А. А. Кухтичев
Группа: М8О-206Б

Дата:
Оценка:
Подпись:

Москва, 2025

Лабораторная работа №8

A. 1 Размен монет

Задача: На первой строке заданы два числа, N и p $>$ 1, определяющие набор монет некоторой страны с номиналами$p^0, p^1, ..., p^{N-1}$. Нужно определить наименьшее количество монет, которое можно использовать для того, чтобы разменять заданную на второй строчке сумму денег $M <= 2^{32} - 1$ и распечатать для каждого i-го номинала на i-ой строчке количество участвующих в размене монет. Кроме того, нужно обосновать почему жадный выбор неприменим в общем случае (когда номиналы могут быть любыми) и предложить алгоритм, работающий при любых входных данных.

1 Описание

Эта задача решается методом жадных алгоритмов. Данный подход предполагает, что алгоритм на каждом шаге делает выбор, который кажется самым лучшим в данный момент — т.е. производится локально оптимальный выбор в надежде, что он приведет к оптимальному решению глобальной задачи.[1]

Алгоритм: Программа на вход получает 3 значения - номинал монеты p, максимальную степень номинала данной монеты -1 n и сумму m, которую необходимо получить из данного набора монет.

Предварительно создадим 2 массива - cnt, в котором будем хранить количество используемых монет каждого номинала, и массив v - в котором мы посчитаем значения всех номиналов монет. Оба массива имеют размерность n.

Т.к. у нас всегда есть монета номиналом $p^0 = 1$, следовательно мы можем получить любую сумму из любого набора монет, чтобы количество монет было минимальным, начнем заполнение нашей суммы с наибольшего наминала монет, и будем вычислять целочисленное деление текущей необходимой суммы монет на номинал текущей рассматриваемой монеты, Если можно добавить монету в нашу сумму (номинал монеты меньше или равен текущей необходимой сумме), то вычитаем из текущей суммы номинал монеты умноженный на результат целочисленного деления суммы на номинал монеты cnt_i и меняем значение cnt[i] на cnt_i.

В противном случае (номинал монеты больше необходимой суммы) переходим к рассмотрению номинала следующей монеты.

Если текущая сумма стала = 0 или мы дошли до минимального номинала (= 1), то цикл завершается и выводится ответ.

Оценка сложности.
Нам необходимо найти все номиналы монет, это делается проходом за цикл длинной n. Затем мы идем по массиву с количеством монет каждого номинала и реализуем шани нашего алгорима, это тоже займет у нас в худшем случае - n итераций. Итоговая сложность - $O(n)$.

Решение задачи для общего случая.
Если рассматривать общий случай, то подход жадного алгоритма будет не применим, т.к. он будет искать локальную оптимизацию, а не глобавльную. В данном случае эту задачу можно интерпретировать как "Задачу о рюкзаке с повторениями которая решается методами динамического программирования. Т.к. динамическое программирование позволяет рассматривать все возможные комбинации предметов, что приводит к оптимальному решению.

2 Исходный код

Данная задача является достаточно простой и переводит на язык программирования весь алгоритм описанный вначале.

```
int main(){
    uint64_t n, p, m;
    cin>>n>>p>>m;
    uint64_t cur_sum=m;
    vector<uint64_t> v(n);
    vector<uint64_t> cnt(n);
    v[0]=1;
    for(uint64_t i=1; i<n;++i){
        v[i]= v[i-1]*p;
    }
    for(uint64_t i=n-1; i>=0;--i){
        uint64_t cnt_i = cur_sum/v[i];
        cnt[i] = cnt_i;
        cur_sum = cur_sum - v[i]*cnt_i;
        if(i==0 or cur_sum==0) break;
    }
    for(uint64_t i=0; i<n;++i){
        cout<<cnt[i]<<'\n';
    }
}
```

3 Консоль

```
arsenii@PC-Larcha14:~/Documents/C_pp_uk/DA/lab_8$ ./lab_8
3 5
71
1
4
2
```

4 Тест производительности

Сравним жадный алгоритм с решением данной задачи методом динамического программирования

```
arsenii@PC-Larcha14:~/Documents/C_pp_uk/DA/lab_8$ g++ banchmark.cpp -o banchmark
arsenii@PC-Larcha14:~/Documents/C_pp_uk/DA/lab_8$ ./banchmark
3 5
71
Dummy algorithm : 0.000008293 s
My algorithm    : 0.000013453 s
arsenii@PC-Larcha14:~/Documents/C_pp_uk/DA/lab_8$ ./banchmark
5 2
100
Greedy algorithm: 0.000008934 s
DP algorithm    : 0.000023982 s
arsenii@PC-Larcha14:~/Documents/C_pp_uk/DA/lab_8$ ./banchmark
10 2
100
Greedy algorithm: 0.000013009 s
DP algorithm    : 0.000028420 s
arsenii@PC-Larcha14:~/Documents/C_pp_uk/DA/lab_8$ ./banchmark
5 2
2000
Greedy algorithm: 0.000010016 s
DP algorithm    : 0.000563381 s
```

Как и ожидалось, жадный алгоритм, выигывает на всех тестах. Жадный алгоритм подходит для задач с определёнными условиями, но может не всегда быть оптимальным. Динамическое программирование всегда гарантирует оптимальное решение, но требует большего времени и памяти.

5 Выводы

Выполнив данную лабораторную работу, я преисполнился в жадных алгоритмов, закрепил знания, полученные о низ ранее, а также, как мне кажется, научился четко различать задачи динамического программирования от жадных алгоритмов. Как и метод динамического программирования, данный подход не применим ко всем задачам, можно даже сказать, что спектр задач решаемых жадным алгоритмом ещё уже чем спектор задач, решаемых методом динамического программирования.

В общем, жадные алгоритмы - мощный инструмент, про который не стоит забывать.

Список литературы

[1] [1] Томас Х. Кормен, Чарльз И. Лейзерсон, Рональд Л. Ривест, Клиффорд Штайн. Алгоритмы: построение и анализ, 2-е издание. — Издательский дом «Вильямс», 2007. Перевод с английского: И. В. Красиков, Н. А. Орехова, В. Н. Романов. — 1296 с. (ISBN 5-8459-0857-4 (рус.))