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. В. Барсов
Преподаватель: Н. С. Капралов
Группа: М8О-307Б-19

Дата:
Оценка:
Подпись:

Москва, 2022

Лабораторная работа №8

Задача: Разработать жадный алгоритм решения задачи, определяемой своим вариантом. Доказать его корректность, оценить скорость и объём затрачиваемой оперативной памяти.

Реализовать программу на языке C или C++, соответсвующую построенному алгоритму. Формат входных и выходных данных описан в варианте задания.

Вариант №2:

На координатной прямой даны несколько отрезков с координатами $[L_i, R_i]$. Необходимо выбрать минимальное количество отрезков, которые бы полностью покрыли интервал $[0, M]$.

Формат входных данных

На первой строке располагается число N, за которым следует N строк на каждой из которой находится пара чисел L_i, R_i; последняя строка содержит в себе число M.

Формат результата

На первой строке число K выбранных отрезков, за которым следует K строк, содержащих в себе выбранные отрезки в том же порядке, в котором они встретились во входных данных. Если покрыть интервал невозможно, нужно распечатать число 0.

1 Описание

Согласно [1], жадные алгоритмы предназначены для решения задач оптимизации. Они обычно представляют собой последовательность шагов, на каждом из которых предоставляется некоторое множестов выборов. В жадном алгоритме всегда делается выбор, который кажется самым лучшим в данный момент, т.е. проводится локально оптимальный выбор в надежде, что он приведет к оптимальному решению глобальой задачи. Важно отметить, что жадные алгоритмы не всегда приводят к оптимальным решениям, но во многих задачах дают нужный результат.

Применим жадный алгортим к задаче выбора отрезков. Сохраним наши пары границ отрезков в массив данных и отсортируем их по правой границе, т.е. при обращении к данному массиву в начале будут лежать отрезки, которые заканчиваются правее всех. Также при сохранении отрезков запомним их изначальный индекс для того чтобы, при выводе восстановить исходный порядок.

Теперь будем считать, что в точке 0, т.е. в начале покрываемого интервала, лежит грань, котрорую мы будем двигать по мере того, как будет происходить покрытие интервала. Эта грань будет означать самую правую точку покрытой части (в начале это точка 0). Пока эта точка не станет >= M, мы будем считать наш интервал непокрытым.

Затем будем в цикле проходиться по нашему отсортированному масиву и брать первый отрезок, левая граница которого лежит левее или равна текущей грани, а правая граница лежит обязательно правее текущей грани. Это необходимые условия, т.к. мы ищем минимальное число отрезков, иначе есть шанс взять отрезки, котрые не подвинут нашу грань. Если же на каком-то этапе прохода мы не нашли такого отрезка, это говорит о том, что мы не пожем покрыть наш интервал данной выборкой отрезков. Подходящие же отрезки будем сохранять в новый массив, который в случае порытия интервала отсортируем по изначальному индексу и выведем в качестве ответа.

Докажем верность данного алгоритма. Благодаря тому, что на каждом шаге мы берем отрезок с самой правой границей, то это будет гаратировать минимальность количества выбранных отрезков. Здесь важно отметить, что жадные алгоритмы дают нам лишь одно оптимальное решение, которых может быть несколько. Следовательно, допустим, что мы имеем оптимальное решение задачи выбора отрезков и на каком-то этапе мы решаем добавить наш отрезок в это решение. У нас есть два варианта:

1) этот отрезок лежит в этом решении, тогда все ок, просто перейдем к следующй подзадаче;

2) этот отрезок не лежит в этом решении, но т.к. согласно нашему алгоритму на данном этапе мы добавляем отрезок с самой правой границей, то он сдвинет грань еще дальше и обязательно будет иметь пересечние со следующим отрезком в оптимальном решении, значит мы можем просто исключить отрезок лежащий в оптимальном

решении на выбранный.

2 Консоль

```
[artem@IdeaPad solution]$ make
g++ -pedantic -Wall -std=c++11 -Werror -Wno-sign-compare -O2 -lm -c main.cpp
g++ -pedantic -Wall -std=c++11 -Werror -Wno-sign-compare -O2 -lm main.o -o
solution
[artem@IdeaPad solution]$ ./solution
3
-1 0
-5 -3
2 5
1
0
[artem@IdeaPad solution]$ python generator.py | ./solution
2
0 6
6 13
[artem@IdeaPad solution]$ python generator.py | ./solution
4
2 5
-1 2
8 11
5 8
```

3 Листинг программы

Для тестирования я реализовал скрит генератор случайных тестов на языке Python: generator.py.

```
from random import *
m = 1000000
min_l, max_l = 5000, 15000
n = 800
print(n)
for i in range(n):
    l = -2*max_l + randint(0, m + 2*max_l)
    print(l, l + randint(min_l, max_l))
print(m)
```

А также скрипт дял проверки корректности решения теста check.py:

```
#
n = int(input())
lr = []
for i in range(n):
    lr.append(list(map(int, input().split())))
m = int(input())
#
a = [0] * m
for l, r in lr:
    for x in range(l, r):
        if 0 <= x < m: a[x] += 1
#
zeros = []
for i in range(len(a)):
    if a[i] == 0: zeros.append(i)
# print(zeros)
#
groups = []
l = zeros[0]
acc = 1
for i in range(len(zeros)-1):
    if zeros[i] != zeros[i+1]-1:
        groups.append((zeros[i]-acc+1, zeros[i]+1))
        acc = 1
    else: acc += 1
groups.append((zeros[-1]-acc+1, zeros[-1]+1))
#
for l, r in groups:
    print(f'[{l} ; {r}]')
```

Ключевая идея алгоритма заключается в этом участке кода, где происходит сортировка отрезков по сложному критерию для дальнеёшего жадного перебора.

```
sort(segs.begin(), segs.end(), [&](const Segment& lhs, const Segment& rhs) {
    return std::tie(lhs.l, rhs.r, lhs.id) < std::tie(rhs.l, lhs.r, rhs.id);
});
```

Сохранение порядка ввода отрезков для вывода их в ответ достигается с помощью заранее сохранённых идентификационных номеров каждого отрезка.

```
sort(cover.begin(), cover.end(), [&](const Segment& lhs, const Segment& rhs) {
    return lhs.id < rhs.id;
});
std::cout << cover.size() - 1 << '\n';
for (int i = 1; i < cover.size(); i++)
    std::cout << cover[i].l << ' ' << cover[i].r << '\n';
```

4 Выводы

Выполнив восьмую лабораторную работу по курсу «Дискретный анализ», я изучил жадные алгоритмы и применил идею жадного подхода для решения задачи выбора наименьшего числа отрезков для покрытия интевала.

В отличие от динамического программирования жадные алгоритмы предполагают, что задача имеет оптимальное решение, которое строится из оптимальных решений для подзадач с заранее определённым выбором. Такой подход уменьшает временные и пространственные ресурсы, необходимые для решения задачи.

Важно понимать, что жадные алгоритмы применимы не ко всем типам задач, иногда одна из двух, на первый взгляд, похожих задач может решаться при помощи жадного алгоритма, а другая нет. Например, задачи о непрерывном (решается при помощи ЖА) и дискретном рюкзаке (решается при помощи ДП).

Жадные алгоритмы часто используются на практике, т.к. в реальном мире часто приходится работать с данными огромного размера, поэтому вычислительных мощностей для точного алгоритма может не хватать. По этой причине применяются приближенные жадные алгоритмы, которые работают гораздо быстрее и дают лишь одно оптимальное решение.

Список литературы

[1] Томас Х. Кормен, Чарльз И. Лейзерсон, Рональд Л. Ривест, КлиффордШтайн. *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.))

[2] *Справочник по алгоритмам и структурам данных.*
URL: `https://e-maxx.ru/` .