Лекция №14
по дисциплине
«ТЕОРИЯ АЛГОРИТМОВ»

СЛОЖНОСТЬ АЛГОРИТМОВ ЗАВЕРШЕНИЕ

Преподаватель:
Золотоверх Д.О.

АЛГОРИТМЫ ОТЛИЧАЮТСЯ

Некоторые алгоритмы можно выполнить за секунды.

Некоторые алгоритмы не выполнимы за все время жизни вселенной.

Один алгоритм при выполнении задачи может использовать всю доступную память, когда другой – при увеличении количества входных данных использует ее постоянное количество.

ОЦЕНКА СЛОЖНОСТИ АЛГОРИТМОВ

Сложность алгоритма зависит от задачи (от ее размера и природы).

Измеряется в количестве работы выполненной алгоритмом:

- количество циклов работы процессора;
- количество времени;
- количество памяти.

При расчете сложности необходимо учитывать размер задачи.

Способ оценки сложности алгоритма

Нам необходимо найти количество операций O;

Количество операций, как было описано раньше, зависит от сложности (размера) задачи *n*;

Иногда в коде происходят действия, количество которых не зависит от размера выполняемой задачи – константные значения;

Как говорилось ранее, подсчет совершается для худшего исхода (перебор всех значений, последнее место в массиве).

АСИМПТОТИКА

Полный расчет сложности алгоритма может быть очень трудоемким.

Поэтому для его расчета применяется так называемое асимптотическое равенство.

Простыми словами – убираются малозначимые значения.

Когда задача становится очень большой – их значения незначимы.

Например:

$O = n + 4$ стремится к $O = n$

$O = \log_2(3n) + 2$ стремится к $\log_2(n)$

ТИПЫ АЛГОРИТМОВ

Существуют следующие типы сложности:

- O(1) – кол-во операций не растет с задачей;
- O(log n) – рост кол-ва замедляется с ростом задачи
- O(n) – рост кол-ва пропорционален росту задачи
- $O(n^2)$, $O(2^n)$, O(n!) – рост кол-ва ускоряется с простом задачи

График роста сложности алгоритмов от роста задачи

ПРИМЕРЫ АЛГОРИТМОВ

Сложность	Название алгоритма	Пример задачи
O(1)	Постоянный	Адресация, работа с хеш-таблицами, Работа с очередями
O(log n)	Логарифмический	Бинарный поиск (отсортированный список) Алгоритмы типа разделяй и властвуй
O(n)	Линейный	Перебор массива, Адресация связанного списка, Сравнение строк,
O(n log n)	Линейно-арифмический	Сортировки типа Merge Sort, Heap Sort, Quick Sort
$O(n^2)$	Квадратичный	Работа с двумерным массивом, Сортировки типа Bubble Sort, Insertion Sort, Selection Sort
$O(n^3)$	Кубический	Решение уравнений с 3 переменными
$O(k^n)$	Экспоненциальный	Нахождение всех подмножеств
O(n!)	Факториальный	Найти все перестановки заданного набора, Задача коммивояжера

ВСЕ ПОДМНОЖЕСТВА МНОЖЕСТВА

Множество – совокупность каких-либо объектов, что являются элементами этого множества.

Подмножество – это понятие части множества.

Необходимо найти все возможные подмножества заданного множества.

Например для: 1, 2, 3 все подмножества следующие:

1; 2; 3;

1, 2; 2, 3;

1, 2, 3;

.

РЕШЕНИЕ

Количество элементов	Количество комбинаций	Количество операций
3	8	53
4	16	109
5	32	221
6	64	445
7	128	893
8	256	1789

```
void powerSet(string str, int index = -1,
              string curr = "")
{
    int n = str.length();
    if (index == n) {
        return;
    }

    cout << curr << endl;

    for (int i = index + 1; i < n; i++) {
        curr += str[i];
        powerSet(str, i, curr);
        curr.erase(curr.size() - 1);
    }
    return;
}
```

ОЦЕНКА СЛОЖНОСТИ

При увеличении роста задачи, алгоритм растет экспоненциально:

$O = 2^n\ 6 + n$ приближается к $O = 2^n$

Таким образом алгоритм очень неэффективен:

если n = 10, O = 7168;

если n = 20, O = 7340032;

если n = 100,

O = 887355420159760581047692243763 2.

ПРОБЛЕМА БАШЕН ХАНОЯ

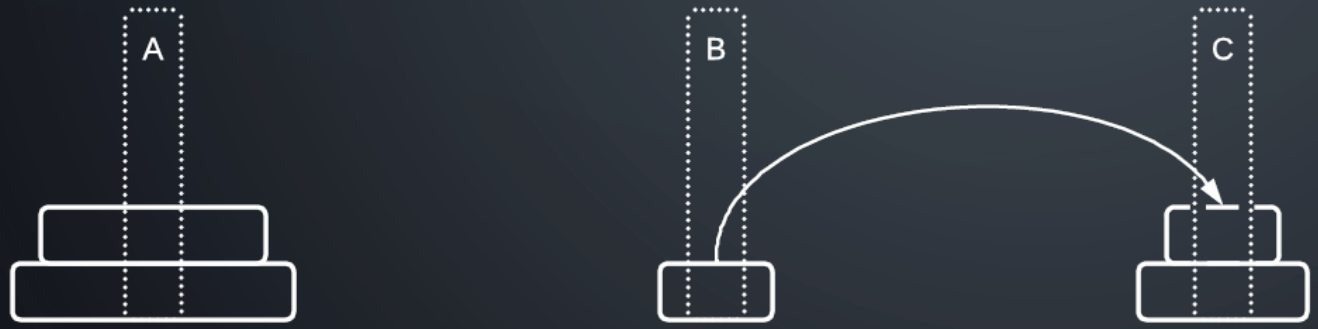
Даны три стержня, на один из которых нанизаны определенное количество колец, причём кольца отличаются размером и лежат меньшее на большем.

Задача состоит в том, чтобы перенести пирамиду из восьми колец за наименьшее число ходов на другой стержень

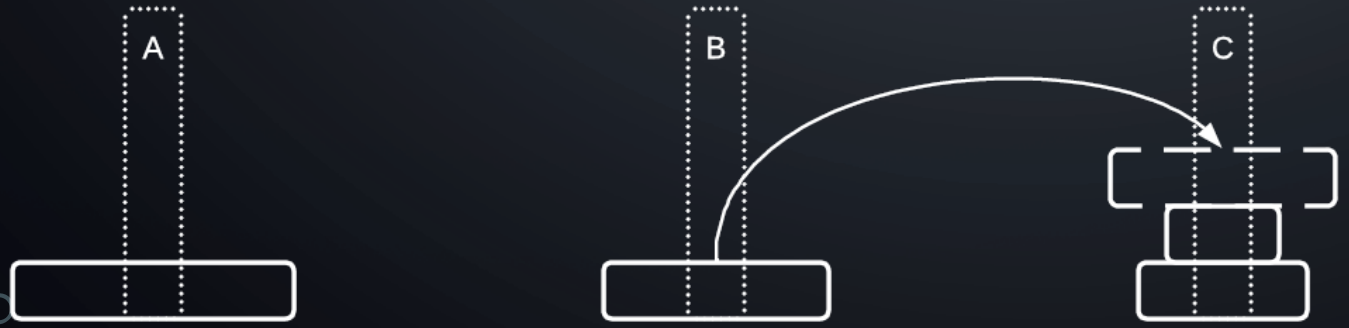
а один раз разрешается переносить только одно кольцо, причём нельзя класть большее кольцо на меньшее.

ПРОБЛЕМА БАШЕН ХАНОЯ

Пример правильного перемещения:



Пример неправильного перемещения:



РЕШЕНИЕ

Количество дисков	Количество перемещений	Количество операций
3	7	29
4	15	61
5	31	125
6	63	253
7	127	509
8	255	1021

```
void towerOfHanoi(
    int n, char from_rod,
    char to_rod, char aux_rod
    ) {
    if (n == 0) {
        return;
    }
    towerOfHanoi(
        n - 1, from_rod, aux_rod, to_rod);
    step++;

    printf(
        "%d)\tMove disk %d from rod %c to rod %c\n",
         step, n, from_rod, to_rod
         );
    towerOfHanoi(
        n - 1, aux_rod, to_rod, from_rod);
}
```

ОЦЕНКА СЛОЖНОСТИ

При увеличении роста задачи, алгоритм растет экспоненциально:

$O = 2^n\ 3 - 1$ приближается к $O = 2^n$

Таким образом алгоритм очень неэффективен:

если n = 10, O = 4093;

если n = 20, O = 4194301;

если n = 100,

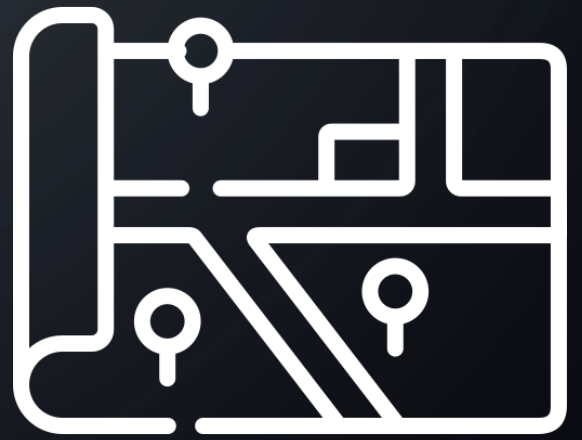
O = 507060240091291760598681282150 1.

ЗАДАЧА КОММИВОЯЖЁРА

Заключающаяся в поиске самого выгодного маршрута, проходящего через все указанные города хотя бы по одному разу с последующим возвратом в исходный город.

Города могут быть соединены, их соединения могут иметь определенное значение (например расстояние).

В условиях задачи указываются критерий выгодности маршрута (кратчайший, самый дешёвый и т.д.)

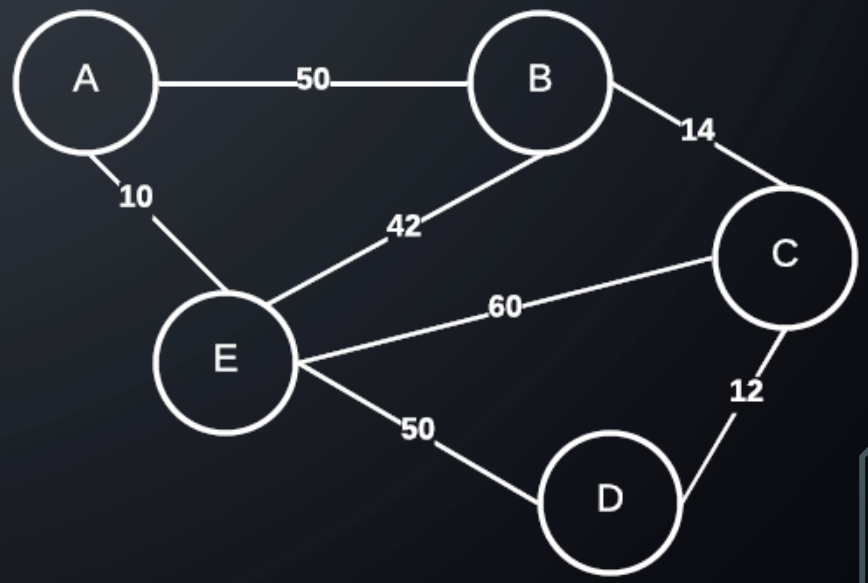


ЗАДАЧА КОММИВОЯЖЁРА

Поскольку коммивояжёр в каждом из городов встает перед выбором следующего города из тех, что он ещё не посетил.

Задача коммивояжёра относится к числу трансвычислительных.

уже при относительно небольшом числе городов (66 и более) она не может быть решена методом перебора.



ОЦЕНКА СЛОЖНОСТИ

В каждом из городов встает перед выбором следующего города из тех, что он ещё не посетил, существует (n – 1)! маршрутов

O = (n – 1)! приближается к O = n!

Таким образом алгоритм очень неэффективен:

если n = 10, O = 362880;

если n = 20, O = 121645100408832000;

если n = 100,

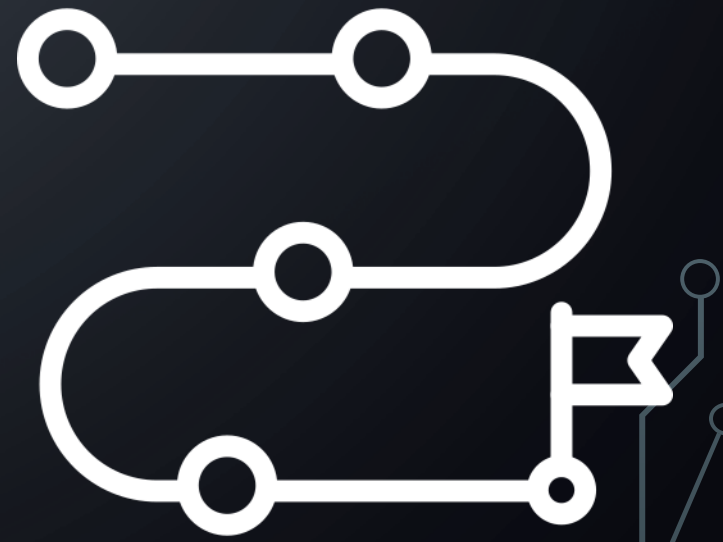
O = 933262154439441526816992388562667004907159682643816214685929638952175999932299156089414639761565182862536979208272237582511852109168640000000000000000000.

ИТОГ

Концепцию O n необходимо понимать, чтобы уметь видеть и исправлять неоптимальный код.

Ни один серьёзный проект, как ни одно серьёзное собеседование, не могут обойтись без вопросов о O n.

Непонимание O n ведёт к серьёзной потере производительности ваших алгоритмов.



СПАСИБО ЗА ВНИМАНИЕ!