THREE REASONS TO STUDY ALGORITHMS

THEORY OF ALGORITHMS

March 26, 2021

1. Predict running time of an algorithm based on input size (Analysis).

How long to sort 300 million names ?

```
1  void insertion_sort(vector<string> &V)
2  {
3    for (int i = 1; i < V.size(); ++i)
4      for (int j = i; j > 0 && V[j] < V[j-1]; --j)
5        swap(V[j], V[j-1]);
6
7  }
```

Answer

Assuming 3 billion comparisons can be compared each second,

$$\frac{(3 \cdot 10^8)(3 \cdot 10^8 - 1)}{(2)(3 \cdot 10^9)(31536000)} \approx 0.47 \text{ years}$$

How long does `f(100)` take ?

```
1  unsigned f(unsigned n)
2  {
3    if (n == 0)
4      return 0;
5    return 1 + f(n-1) + f(n-1);
6  }
```

Answer

$$\frac{2^{100}}{(3 \cdot 10^9)(31536000)(14 \cdot 10^9)} \approx 957 \text{ times universe age}$$

Which algorithm is faster ?

Try: a = 3210987654; b = 2109876543

```
1   uint gcd1(uint a, uintb)
2   {
3     for (uint d = min(a, b); d > 0; --d)
4       if (a % d == 0 && b % d == 0)
5         return d;
6     return max(a, b);
7
8
9
10  }
```

```
1   uint gcd2(uint a, uint b)
2   {
3     while (b > 0)
4     {
5       uint temp = b;
6       b = a % b;
7       a = temp;
8     }
9     return a;
10  }
```

2. Given a problem, design an algorithm to solve it.

Design an algorithm for this problem

INPUT: a positive integer N;

OUTPUT: $\lfloor \sqrt{N} \rfloor$.

Answer

Want: **largest integer** whose square is at most N.

Exhaustively try all candidates between 1 and N.

```
1  int exhaustive_search(int N)
2  {
3    for (int i = 1; i <= N; ++i)
4      if (i*i > N)
5        return i-1;
6
7  }
```

Design an algorithm for this problem

INPUT: integers b and e;

OUTPUT: b^e.

Answer

Use recursion:

$$b^e = \begin{cases} (b^{(e/2)})^2 & \text{if } e \text{ is even} \\ b \cdot b^{e-1} & \text{otherwise.} \end{cases}$$

```
1  int divide_conquer(int b, int e)
2  {
3    if (e == 0)
4      return 1;
5
6    if  (e % 2 == 0)
7    {
8      int temp = divide_conquer(b, e/2);
9      return temp * temp;
10   }
11   else
12     return b * divide_conquer(b, e-1);
13 }
```

Design an algorithm for this problem

INPUT: integers a and b;

OUTPUT: the largest common multiple of a and b.

Answer

Transform problem to another problem whose solution is known:

$$lcm(a, b) = \frac{ab}{gcd(a, b)}.$$

```
1  int transform_conquer(int a, inb )
2  {
3    return (a*b)/gcd(a, b);
4  }
```

Design an algorithm for this problem

INPUT: integer N, and k coin denominations;

OUTPUT: smallest number of coins in given denominations to make change for amount N.

Answer

Greedy

- repeatedly take a coin of largest possible denomination;
- fast but does **not** always give the correct answer.

Dynamic programming

- solve problem when just **one** type of coin is available;
- use this solution to solve problem when **two** types of coin are available;
- repeatedly relax the restriction to three, four, etc, until the original problem is solved (bootstrapping).

3. Determine the minimum time required to solve a given problem

Sorting

We know many sorting algorithms: insertion, selection, merge, heap, quick.

Is there a faster solution not yet discovered ?

Answer

merge sort/heap sort is the fastest possible sorting algorithm (up to a multiplicative constant, and if only comparisons are allowed.)

Coin-change problem

The general (dynamic-programming) solution to the coin-change problem is slow.

Is there a faster solution not yet discovered ?

Answer

The coin-change problem is **NP-hard**, so it is unlikely that a significant faster solution exists.