Basic Techniques: Recursion, Repeating, and Geometric Series

Yufei Tao

Department of Computer Science and Engineering
Chinese University of Hong Kong

Today we will discuss three basic techniques of algorithm design:

- Recursion
- Repeating (till success)
- Geometric Series.

Recursion

Principle of recursion

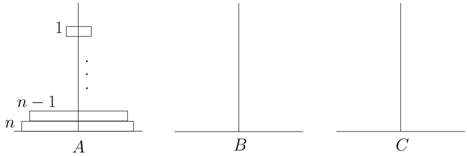
> When dealing with a subproblem (same problem but with a smaller input), consider it solved, and use the subproblem's output to continue the algorithm design.

Tower of Hanoi

There are 3 rods A, B, and C.

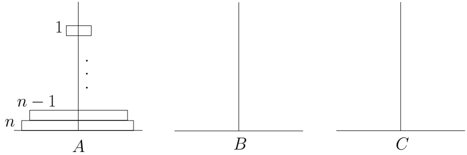
On rod A, *n* disks of different sizes are stacked in such a way that no disk of a larger size is above a disk of a smaller size.

The other two rods are empty.



Tower of Hanoi

Permitted operation: Move the top-most disk of a rod to another rod.
Constraint: No disk of a larger size can be above a disk of a smaller size.

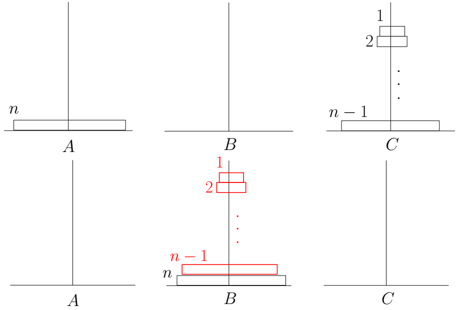


Goal: Design an algorithm to move all the disks to rod B.

Algorithm

Subproblem: Same problem but with $n - 1$ disks.
Consider the subproblem solved (i.e., assume you already have an algorithm for it).

Now, solve the problem with n disks as follows:



Analysis

Suppose that our algorithm performs $f(n)$ operations to solve a problem of size n. Clearly, $f(1) = 1$. By recursion, we can write

$$f(n) \leq 1 + 2 \cdot f(n-1)$$

Solving this recurrence gives $f(n) \leq 2^n - 1$.

Use recursion to "redesign" the following algorithms:

- Binary search
- Quick sort

Repeating till Success

The k-Selection Problem: You are given a set S of n integers in an array and an integer $k \in [1, n]$. Find the k-th smallest integer of S.

For example, suppose that $S = (53, 92, 85, 23, 35, 12, 68, 74)$ and $k = 3$. You should output 35.

The **rank** of an integer $v \in S$ is the number of elements in S smaller than or equal to v.

For example, suppose that $S = (53, 92, 85, 23, 35, 12, 68, 74)$. Then, the rank of 53 is 4, and that of 12 is 1.

Easy: The rank of v can be obtained in $O(|S|)$ time.

Consider the following task:

Task: Assume n to be a multiple of 3. Obtain a subproblem of size at most $2n/3$ with exactly the same result as the original problem.

Our goal is to produce a set S' and an integer k' such that

- $|S'| \leq 2n/3$
- $k' \in [1, |S'|]$
- The element with rank k' in S' is the element with rank k in S.

We will give an algorithm to accomplish the task in $O(n)$ expected time.

Consider the following algorithm.

1. Take an element $v \in S$ uniformly at random.
2. Divide S into S_1 and S_2 where
   - S_1 = the set of elements in S less than or equal to v;
   - S_2 = the set of elements in S greater than v.
3. If $|S_1| \geq k$, then return $S' = S_1$ and $k' = k$;
   else return $S' = S_2$ and $k' = k - |S_1|$.

The algorithm **succeeds** if $|S'| \leq 2n/3$, or **fails** otherwise.

Repeat the algorithm until it succeeds.

Lemma: The algorithm succeeds with probability at least $1/3$.

Proof: The algorithm always succeeds when the rank of v falls in $[\frac{n}{3}, \frac{2}{3}n]$ (think: why?). This happens with a probability at least $1/3$, by the fact that v is taken from S uniformly at random. □

In general, if an algorithm succeeds with a probability **at least** $c > 0$, then the number of repeats needed for the algorithm to succeed for the first time is **at most** $1/c$ in expectation.

We expect to repeat the algorithm at most 3 times before it succeeds. This implies that the expected running time is $O(n)$ (think: why?).

Geometric Series

A **geometric sequence** is an infinite sequence of the form

$$n, cn, c^2 n, c^3 n, \ldots$$

where n is a positive number and c is a constant satisfying $0 < c < 1$.

It holds in general that

$$\sum_{i=0}^{\infty} c^i n = \frac{n}{1-c} = O(n).$$

The summation $\sum_{i=0}^{\infty} c^i n$ is called a **geometric series**.

Geometric series are extremely important for algorithm design.

Consider again:

The k-Selection Problem: You are given a set S of n integers in an array and an integer $k \in [1, n]$. Find the k-th smallest integer of S.

Algorithm

Using the repeating technique, now you should be able to convert the problem to a subproblem with size at most $\lceil 2n/3 \rceil$ in $O(n)$ expected time.

Now, apply the recursion technique. We have already obtained a (complete) algorithm solving the k-selection problem!

Think: How is this related to geometric series?

Analysis

Let $f(n)$ be the expected running time of our algorithm on an array of size n.

We know:

$$
\begin{aligned}
f(1) &\leq O(1) \\
f(n) &\leq O(n) + f(\lceil 2n/3 \rceil).
\end{aligned}
$$

Solving the recurrence gives $f(n) = O(n)$.