CSC 212: Data Structures and Abstractions

13: Analysis of Recursive Algorithms

Prof. Marco Alvarez

Department of Computer Science and Statistics
University of Rhode Island

Spring 2025

THINK BIG WE DO℠

Recurrence relations

- Recurrence relation
  - a recurrence is an equation that expresses each element of a sequence as a function of the preceding ones
  - e.g., $T(n) = T(n-1) + 1$
- Recurrences and algorithm analysis
  - recurrence relations are used to analyze the time complexity of recursive algorithms
  - the time complexity of a recursive algorithm is the solution to the recurrence relation that describes the cost of the algorithm
  - exact closed-form solution may not exist, or may be too difficult to find
  - for most recurrences, an asymptotic solution of the form $\Theta()$ is acceptable in the context of analysis of algorithms

Solving recurrence relations

- Recurrences can be solved using a variety of techniques
  - **substitution method**: guess the form of the solution and prove it by induction
  - **recursion tree method**: draw a recursion tree and sum the costs at each level
  - **master theorem**: a general method for solving recurrences of the form $T(n) = aT(n/b) + f(n)$
  - **unrolling method**: expand the recurrence into a series of equations and solve them
- Unrolling (iteration) method
  - expand the recurrence until you identify a general case, then use the base case
  - not trivial in all cases but it is helpful to build an intuition
  - induction may be necessary to prove correctness

Practice

$$T(0) = 0$$

$$T(n) = T(n-1) + 1$$

```
double power(double b, int n) {
    // base case
    if (n == 0) {
        return 1;
    }
    // recursive call
    return b * power(b, n-1);
}
```

Analysis of binary search

- Base case: $T(0) = c_0$
- Recursive case: $T(n) = T(n/2) + c_1$

```
int bsearch(std::vector<int>& A, int lo, int hi, int k) {
    // base case
    if (hi < lo) {
        return -1;
    }
    // calculate midpoint index
    int mid = lo + ((hi-lo)/2);
    // key found?
    if (A[mid] == k)
        return mid;
    // key in upper subarray?
    if (A[mid] < k)
        return bsearch(A, mid+1, hi, k);
    // key is in lower subarray?
    return bsearch(A, lo, mid-1, k);
}
```


Analysis of binary search

- Solve: $T(0) = c_0, \quad T(n) = T(n/2) + c_1$

Practice

$$T(1) = 1$$
$$T(n) = 2T(n/2) + n$$

Practice

$$T(0) = 1$$
$$T(n) = 2T(n-1) + 1$$

Series

$$\sum_{i=1}^{n} i = \frac{n(n+1)}{2}, \quad \sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6}, \quad \sum_{i=1}^{n} i^3 = \frac{n^2(n+1)^2}{4}.$$

In general:

$$\sum_{i=1}^{n} i^m = \frac{1}{m+1}\left[(n+1)^{m+1} - 1 - \sum_{i=1}^{n}\left((i+1)^{m+1} - i^{m+1} - (m+1)i^m\right)\right]$$

$$\sum_{i=1}^{n-1} i^m = \frac{1}{m+1}\sum_{k=0}^{m}\binom{m+1}{k} B_k n^{m+1-k}.$$

Geometric series:

$$\sum_{i=0}^{n} c^i = \frac{c^{n+1}-1}{c-1}, \quad c \neq 1, \quad \sum_{i=0}^{\infty} c^i = \frac{1}{1-c}, \quad \sum_{i=1}^{\infty} c^i = \frac{c}{1-c}, \quad |c| < 1,$$

$$\sum_{i=0}^{n} ic^i = \frac{nc^{n+2} - (n+1)c^{n+1} + c}{(c-1)^2}, \quad c \neq 1, \quad \sum_{i=0}^{\infty} ic^i = \frac{c}{(1-c)^2}, \quad |c| < 1.$$

Harmonic series:

$$H_n = \sum_{i=1}^{n} \frac{1}{i}, \qquad \sum_{i=1}^{n} iH_i = \frac{n(n+1)}{2}H_n - \frac{n(n-1)}{4}.$$

$$\sum_{i=1}^{n} H_i = (n+1)H_n - n, \quad \sum_{i=1}^{n}\binom{i}{m}H_i = \binom{n+1}{m+1}\left(H_{n+1} - \frac{1}{m+1}\right).$$