1 Further Analysis of Algorithms

Consider the following function:

function square(n)

```
if n = 1
  then return n
   else return(n + n − 1 + square(n − 1))
fi
```

Define $T : \mathbb{Z}^+ \to \mathbb{N}$, where $T(n) =$"the number of arithmetic operations performed by the square(n)".

Then $T(n) = \begin{cases} 0, n = 1 \\ 4 + T(n-1), n > 1 \end{cases}$.

Now, consider the mergesort algorithm:

MERGESORT(A, n)

```
if n = 1 then return
            divide A into 2 subarrays A′ and A″ of size ⌈n/2⌉ and ⌊n/2⌋
            MERGESORT(A′, ⌈n/2⌉)
            MERGESORT(A″, ⌊n/2⌋)
fi
A ← MERGE(A′, A″)
return
```

For $n \in \mathbb{Z}^+$, let $M(n) =$"the worst case time complexity of MERGESORT(A, n) over all arrays A of size n".

Then $M(n) = \begin{cases} c, n = 1 \\ M(\lceil n/2 \rceil) + M(\lfloor n/2 \rfloor)) + dn, n > 1 \end{cases}$, where c, d are constants.

Consider now the binary search algorithm.

BINSEARCH(A, f, l, x)

```
if f = l then
           if A[f] = x then returnf
                      else return 0
           fi
fi
m ← ⌊f+l/2⌋
if A[m] ≥ x then
              return BINSEARCH(A, f, m, x)
           else returnBINSEARCH(A, m + 1, l, x)
```

Define $B : \mathbb{Z}^+ \to \mathbb{N}$, where $B(n)$ denotes the wors case number of comparisons with x performed by BINSEARCH(A, f, l, x), where $n = lsf + 1$ over all choices of A, f, l, x.

Then $B(n) = \begin{cases} 1, n = 1 \\ 1 + \max\{B(\lceil n/2 \rceil), B(\lfloor n/2 \rfloor)\}, n > 1 \end{cases}$

2 Methods of Solving Recurrences

2.1 Guess and Verify

- genereate a table of values
- look for a pattern
- guess a solution
- prove it is a solution by induction

2.2 Repeated Substitution / Plug & Chug

To find a closed form for $M(n)$ from above, consider the special case where n is a power of 2.

Then $M(n) = \begin{cases} c, n = 1 \\ 2M(\frac{n}{2}) + dn, n > 1 \end{cases}$.

Therefore,

$$\begin{aligned} M(n) &= 2M(n/2) + dn && (1) \\ &= 2[2M(n/4 + dn/2] + dn && (2) \\ &= 4M(n/4) + 2dn && (3) \\ &= 2^i M(n/2^i) + idn. && (4) \end{aligned}$$

For $k \in \mathbb{N}$, let $Q(k) =$"$M(2^k) = c2^k + dk2^k$".

Prove that $\forall k \in \mathbb{N}.Q(k)$.

Theorem 2.1
$M(n) \in O(n \log n)$.

Proof:

To prove this, we show that $M(n)$ is a nondecreasing function $M(n+k) \geq M(n)$ for all $n \in \mathbb{Z}^+$, $k \in \mathbb{N}$.

Let $n \in \mathbb{Z}^+$ be arbitrary. Let 2^k be the smallest power of two that is greater than or equal to n, i.e. $k = \lceil \log_2 n \rceil$.

Since M is nondecreasing, then $n \leq 2^k \leq 2n$.

Thus, $M(n) \leq M(2^k) = c2^k + dk2^k < c2n + d2n \log(2n) = 2cn + 2dn(\log_2 n + 1)$.

Hence, $\forall n \in \mathbb{Z}^+.M(n) \leq 2cn + 2dn \log_2 n + 2dn$, and so $M(n) \in O(n \log n)$.

Lemma 2.2
$\forall m \in \mathbb{Z}^+.\forall n \in \mathbb{Z}^+.[m \leq n \text{ then } M(m) \leq M(n)]$.

Proof:

For $n \in \mathbb{Z}^+$, let $R(n) =$"$\forall.m \in \mathbb{Z}^+.m \leq n \text{ then } M(m) \leq M(n)$".

Let $n \in \mathbb{Z}^+$ be arbitrary and suppose that $R(n')$ is true for all $n' \in \mathbb{Z}^+$ such that $n' < n$.

Then $M(1) = c < 2c + 2d = M(2)$. Therefore, $R(1)$ and $R(2)$ are true.

Now consider $n > 2$. Then $1 \leq \lfloor n/2 \rfloor \leq \lceil n/2 \leq n - 1 < n \rceil$.

Note that $R(\lfloor n/2 \rfloor)$, $R(\lceil n/2 \rceil)$ and $R(n-1)$ are all true by inductive hypothesis.

Let $M \in \mathbb{Z}^+$ be arbitrary. Suppose $m \leq n$.

If $m = n$, then $M(m) = M(n)$ by substitution.

Suppose then $m = n - 1$.

Exercise: continue the proof. And look up the Master Theorem.