CS3230 Chapter 2 - Simple Sorting and Searching

Based on lectures by Chang Ee-Chien
Notes taken by Andrew Tan
AY18/19 Semester 1

1 Simple Sorting

Sorting algorithms are a group of algorithms that puts elements of a given input list in a certain order.
For analysis of sorting algorithms, the elementary operations obtained are the number of comparisons and the number of swaps made, with the input size being the number of elements to be sorted.
As the data being sorted may require a relatively large amount of resources to store, we primarily consider only the comparison and movement of this data, as opposed to other operations on control variables.

1.1 Selection Sort

Selection sort repeatedly moves the smallest element from the input to the front.

```
for i ← 1 to N − 1 do
   min ← i;
   for j ← i + 1 to N do
      if a[j] < a[min] then min ← j;
   t ← a[min];
   a[min] ← a[i];
   a[i] ← t;
end of for-loop;
```

Complexity:
For comparisons, Inner loop performs $n - i$ comparisons. Outer loop performs $n - 1$ loops. So,

$$\sum_{i=i}^{n-1} n - 1 = \frac{n^2-n}{2} \in O(n^2)$$

For data movements: Number of data movements: $(n - 1)$
Thus, number of comparisons $\in \Theta(n^2)$, and
number of data movements $\in \Theta(n)$.

1.2 Insertion Sort

Insertion sort repeatedly picks an element from the input, and inserts it into its sorted position in the output subsequence.

```
for i ← 2 to n do
   v ← a[i];
   j ← i
   while a[j − 1] > v do
      a[j] ← a[j − 1];
      j ← j − 1;
   end of while-loop;
a[j] ← v;
end of for-loop;
```

Correctness can be proven through mathematical induction, by considering the output subsequence before and after insertion.

Complexity:
The while-loop itself is executed a variable number of times depending on the input instance.

Consider the worst and best cases:
Worst case is if the input instance is in decreasing order, hence insertion sort will take at least $2 + 3 + ... + (n - 1)$ comparisons.
Best case is if the input instance is already sorted, so only one pass is needed through the input.
Hence, number of comparisons in the worst case $\in \Omega(n^2)$, and
number of comparisons in the best case $\in O(n)$.
Also, number of data-movements in the worst case $\in \Theta(n^2)$

1.3 Lower Bound

Other sorting algorithms such as merge-sort and or heap-sort have an average and/or best case of $\Omega(n \log(n))$

Comparison based sorting algorithms have a **lower bound** of $\Omega(n \log(n))$

2 Searching

A search algorithm is any algorithm which retrieves information stored within some data structure or calculated in the search space of a problem domain.

2.1 Binary Search

Binary search is a search algorithm that utilizes a divide-and-conquer approach to search for a value within a sorted input.

2.1.1 Recursive implementation

```
BinarySearch(A, key, l, r):
   if l > r return NOT FOUND;
   m ← ⌊(l+r)/2⌋;
   if key == A[m] return m;
   if key < A[m] return BinarySearch(A, key, l, m − 1);
   if key > A[m] return BinarySearch(A, key, m + 1, r);
```

2.1.2 Iterative implementation

```
BinarySearch(A, key):
l ← 1; r ← n;
while r ≥ l:
   m ← ⌊(l+r)/2⌋;
   if A[m] < key then l ← m + 1;
   if A[m] > key then r ← m − 1;
   if A[m] == key then return m;
end of while-loop;
return NOT FOUND;
```

3 Additional Math Addendums

3.1 Little-o

Let $f(n)$ and $g(n)$ be functions on $\mathbb{Z}_+$. We say that $f(n) \in o(g(n))$ if for any positive constants c, there exists positive constants n_0 such that $\forall n > n_0, f(n) \leq cg(n)$.
So, if $f(n) \in o(g(n))$, $f(n) \in O(g(n))$ and $f(n) \notin \Omega(g(n))$

3.2 Approximation

Often, we use $\approx$ to denote that if $f(n) \approx g(n)$ we can use $g(n)$ as an approximation of $f(n)$ in subsequent analysis without loss of generality.
Thus, $f(n) \approx g(n)$ means that $f(n) = g(n) + o(g(n))$

3.3 Mathematical identities and formulae

3.3.1 Sterling approximation

1. $\log_2(n!) = n\log_2(n) - n\log_2(e) + O(\log_2(n))$
2. $n! = \sqrt{2\pi n}(\frac{n}{e})^n(1 + \Theta(\frac{1}{n}))$
3. $\sqrt{2\pi}n^{n+\frac{1}{2}}e^{-n} \leq n! \leq en^{n+\frac{1}{2}}e^{-n}$

3.3.2 Series

1. $\sum_{i=0}^{n} x^i = 1 + x + x^2 + ... + x^n = \frac{1-x^{n+1}}{1-x}$ when $x \neq 1$
2. $\sum_{i=0}^{\infty} x^i = 1 + x + x^2 + ... = \frac{1}{1-x}$ for $|x| < 1$
3. $\sum_{k=0}^{n} \frac{1}{k} = 1 + \frac{1}{2} + ... + \frac{1}{n} = \ln(n) + O(1)$