Algorithm efficiency and asymptotic analysis: best case, worst case, average case

1. Best case, worst case, average case
2. Binary search
3. Insertion sort

1. Best case, worst case, average case

Suppose algorithm *X* exists and *X* has input data whose size is *n*. Suppose the running time function of *X* is *f*(*n*).

Q: With regard to the input data, what is the best case for algorithm *X* ?

A: The best case input data is that for which the asymptotic growth of *f*(*n*) is as least as small as that of any other input data.

Q: With regard to the input data, what is the worst case for algorithm *X* ?

A: The worst case input data is that for which the asymptotic growth of *f*(*n*) is as least as large as that of any other input data.

For best cases and the worst cases one should always be able to provide examples.

Focus on worst case (usually):

- gives a guarantee on how bad *f*(*n*) can be

- worst case can often occur e.g. searching for data that does not occur

Average case is more difficult

- there is no one average case, rather proabilities are associated with input data

- requires probabilitistic analysis to obtain asymptotic results

- often must make assumptions on probability distribution of input data

2. Binary search

Iterative pseudocode for binary search

```
1   BINSEARCHI(A, L, R, key)
2   input A[1..n] is an array containing numbers; L and R are the
3   leftmost and rightmost indices that concern us; key is the value
4   we want to find
5   output: the index at which key occurs A or −1
6   /* The numbers in A must be in order from smallest to largest.
7   We search in array A[L..R] for key. If key is found, we return
8   its location, otherwise we return −1.*/
9   while L < R
10    mid = ⌊(L + R)/2⌋
11    if key == A[mid] then
12      return mid
13    else
14      if A[mid] > key then
15        R = mid
16      else
17        L = mid + 1
18      end
19    end
20  end
21  return −1
```

Input data assumptions:

- sorted input array $A[1..n]$ from smallest to largest

- $L = 1$, $R = n$ and $n > 1$.

Q: What is best case input data?

A: Line 9 **while**-loop is only executed once.

When possible: first iteration of **while**-loop results in $A[mid] = key$

Q: Is best case possible?

A: Yes.

$A =$	−21	−20	−18	−6	5	12	13

key = -6

Running time function for best case:

$f(n) =$ c, where c is a positive constant

Big-Omega and Big-Oh for best case? $O(1)$, $\Omega(1)$

Big-Theta for best case? $\Theta(1)$

Q: What is worst case input data?

A: Line 9 **while**-loop is executed as many times as necessary, until *L* >= *R*.
Let *k* be necessary number of iterations.

n	4	6	8	12	16	120	1000
_ k until *L* >= *R*	2	3	3	4	4	7	10

$$\frac{n}{2^k} = 1 \Rightarrow 2^k = n \Rightarrow k = \log_2 n$$

Q: Is worst case possible?

A: Yes.

e.g. *key* < *A*[1]

Running time function for worst case:

f(*n*) = $c \log_2 n$

Big-Omega and Big-Oh for worst case? $O(\log_2 n)$, $\Omega(\log_2 n)$

Big-Theta for worst case? $\Theta(\log_2 n)$

3. Insertion sort

Pseudocode

```
1   INSERTSORT(A)
2   input: number array A output: sorted array A
3   /* The numbers in input A[1..n] may be in any order. On output the
4   numbers in A are sorted from smallest to largest. */
5   for j from 2 to A.length
6     key = A[j], k = j
7     while k ≥ 2 and A[k − 1] > key
8       A[k] = A[k − 1], k = k − 1
9     end
10    A[k] = key
11  end
```

$n = A.\text{length}$

Previously for entire procedure we obtained lower bound and upper bound on running time:

$f_L = 8n - 8$ $\qquad$ $f_U = 4n^2 + n - 9$

Q: What is best case input data?

A: Case where *f*(*n*) is *fL*.

Q: Is best case possible?

A: Yes.

If input array *A*[1..*n*] is already sorted from smallest to largest.

Big-Omega and Big-Oh for best case? Big-Theta for best case?

$O(n)$, $\Omega(n)$, $\Theta(n)$

Q: What is worst case input data?

A: Case where $f(n)$ is fU.

Q: Is worst case possible?

A: Yes.

If input array $A[1..n]$ is in reverse order from largest to smallest.

Big-Omega and Big-Oh for worst case? Big-Theta for worst case?

$O(n^2)$, $\Omega(n^2)$, $\Theta(n^2)$