

RUNNING TIME ANALYSIS - PART 2

BINARY SEARCH TREES RUNNING TIME

Problem Solving with Computers-II

The image shows the C++ logo in a large, blue, 3D-style font. Below the logo is a snippet of C++ code in a monospaced font, with some words highlighted in color (red for preprocessor directives, blue for keywords, green for strings, and black for other identifiers and punctuation).

```
#include <iostream>
using namespace std;

int main(){
    cout<<"Hola Facebook\n";
    return 0;
}
```

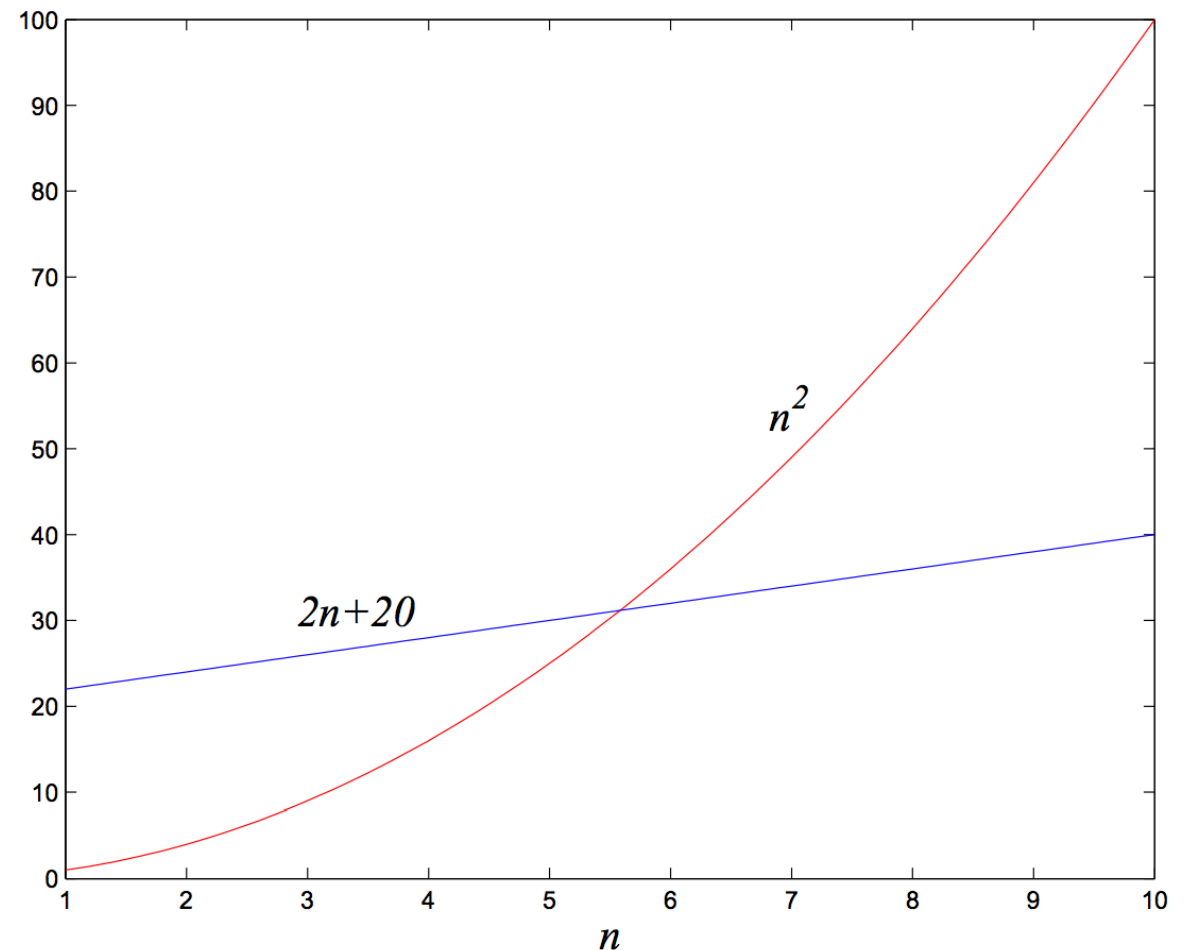
What does Big-Oh really mean?

Formal definition of Big-O

- $f(n)$ and $g(n)$: running times of two algorithms on inputs of size n .
- $f(n)$ and $g(n)$ map positive integer inputs to positive reals.

We say $f = O(g)$ if there is a constant $c > 0$ and $k > 0$ such that
 $f(n) \leq c \cdot g(n)$ for all $n \geq k$.

$f = O(g)$
means that “ f grows no faster than g ”

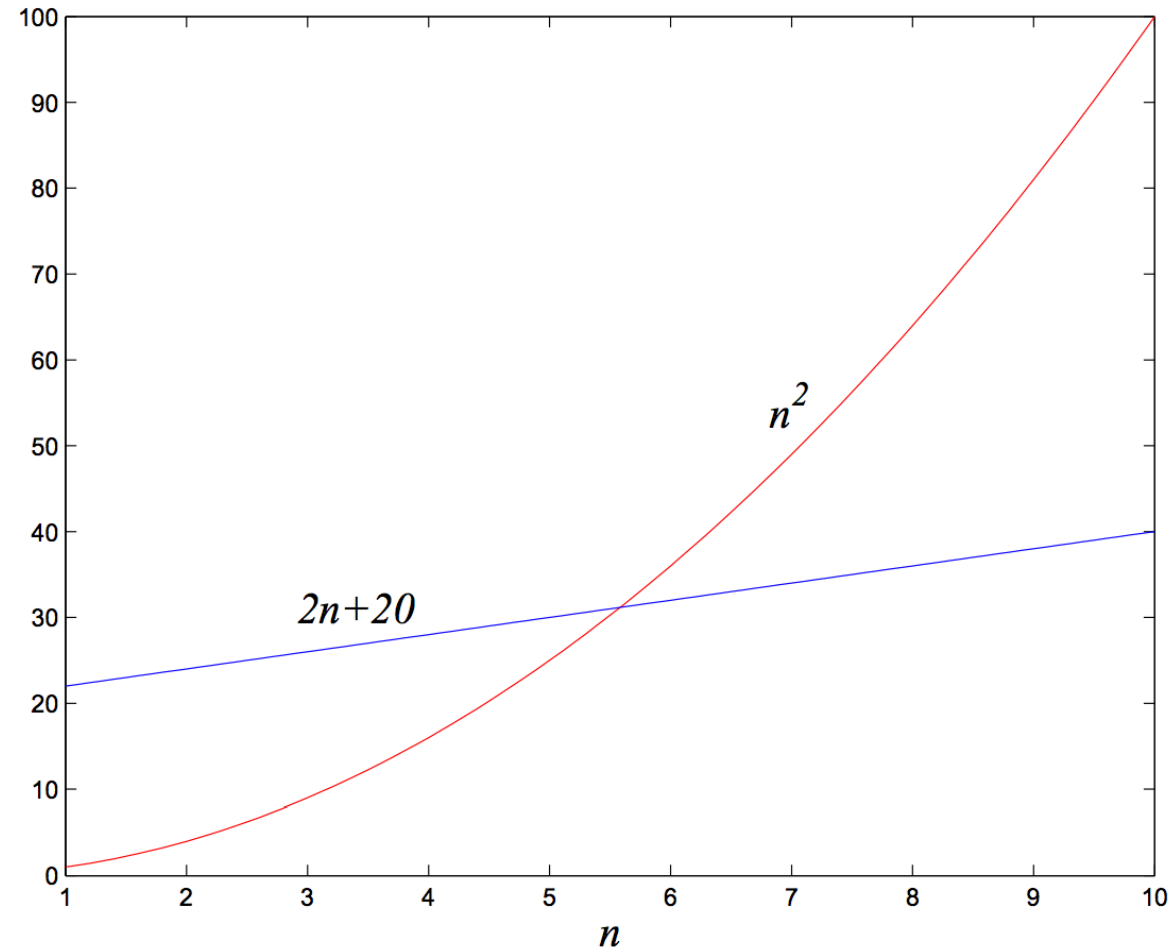


Big-Omega

- $f(n)$ and $g(n)$: running times of two algorithms on inputs of size n .
- $f(n)$ and $g(n)$ map positive integer inputs to positive reals.

We say $f = \Omega(g)$ if there are constants $c > 0$, $k > 0$ such that $c \cdot g(n) \leq f(n)$ for $n \geq k$

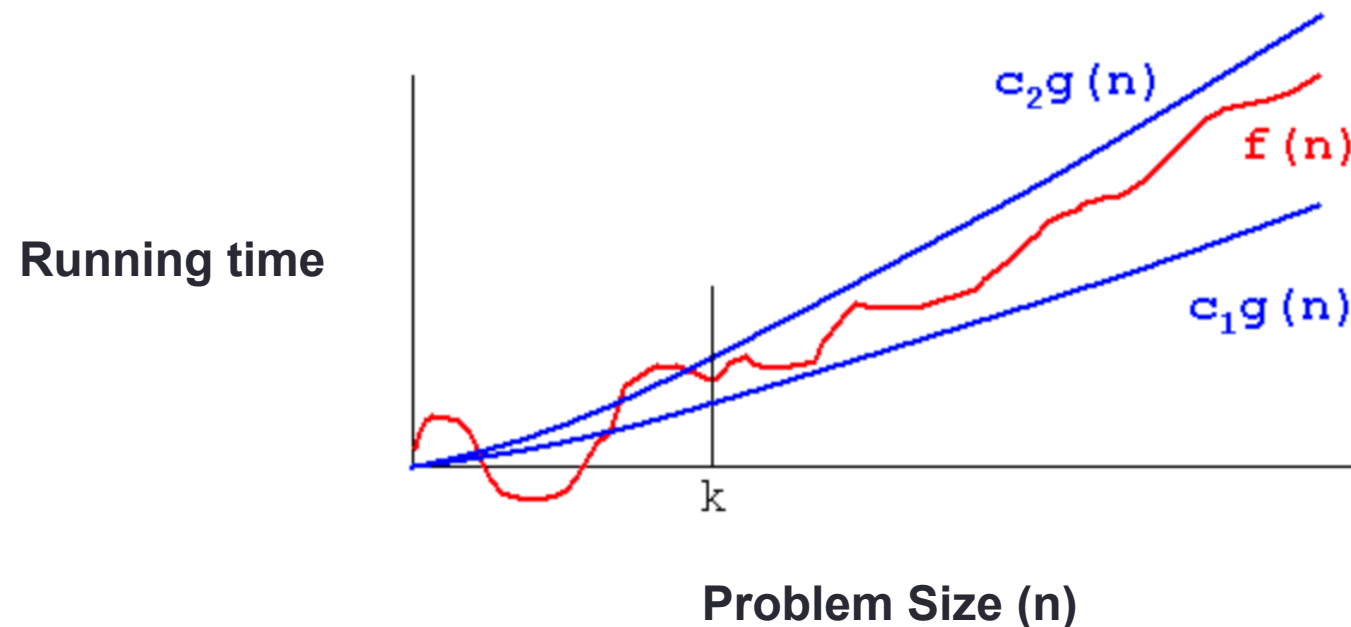
$f = \Omega(g)$
means that “ f grows at least as fast as g ”



Big-Theta

- $f(n)$ and $g(n)$: running times of two algorithms on inputs of size n .
- $f(n)$ and $g(n)$ map positive integer inputs to positive reals.

We say $f = \Theta(g)$ if there are constants c_1, c_2, k such that $0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n)$, for $n \geq k$



What is the Big-O running time of algoX?

- Assume **dataA** is some data structure.
- Given: running time of operations for dataA, where M is the number of keys stored in dataA
 - insert: $O(\log M)$
 - min: $O(1)$
 - delete: $O(\log M)$

```
void algoX(int arr[], int N)
{
    dataA ds; // ds contains no keys
    for(int i=0; i < N; i++)
        ds.insert(arr[i]);
    for(int i=0; i < N; i++){
        arr[i] = ds.min();
        ds.delete(arr[i]);
    }
}
```

- A. $O(N^2)$
- B. $O(N \log N)$
- C. $O(N)$
- D. $O(\log N)$
- E. Not enough information to compute

Best case, worst case, average case running times

Operations on sorted arrays

- Min :
- Max:
- Median:
- Successor:
- Predecessor:
- Search:
- Insert :
- Delete:

[illegible]

Worst case analysis of binary search

```
bool binarySearch(int arr[], int element, int N){  
    //Precondition: input array arr is sorted in ascending order  
    int begin = 0;  
    int end = N-1;  
    int mid;  
    while (begin <= end){  
        mid = (end + begin)/2;  
        if(arr[mid]==element){  
            return true;  
        }else if (arr[mid]< element){  
            begin = mid + 1;  
        }else{  
            end = mid - 1;  
        }  
    }  
    return false;  
}
```