
DSC 40B - Discussion 02

Problem 1.

Suppose $A(n), B(n), C(n)$ are functions describing the runtime of three algorithms. Furthermore, suppose we have the following bounds on each function:

$$\begin{aligned}A(n) &= \Theta(n^3) \\ B(n) &= O(n^2 \log n) \\ C(n) &= O(n^5) \text{ and } C(n) = \Omega(n^2)\end{aligned}$$

What are the best bounds that can be placed on the following functions?

For this problem, you do not need to show work.

Example 1: $A(n) + B(n)$.

Solution: $A(n) + B(n)$ is $\Theta(n^3)$.

Example 2: $f(n) = 2 \cdot C(n)$.

Solution: $f(n) = 2 \cdot C(n)$ is $O(n^5)$ and $\Omega(n^2)$.

- a) $A(n) + B(n)$
- b) $B(n) + C(n)$
- c) $A(n) + C(n)$

Problem 2.

Compute the best and worst case time complexity for the following code snippets:

```
a) def f_1(arr1, arr2):
    """`arr1` and `arr2` are two arrays each of size n"""
    n = len(arr1)
    for i in range(n):
        for j in range(n):
            if arr1[i] + arr2[j] == 0:
                return (i, j)

b) def insertion_sort(arr):
    """Sort `arr` in ascending order."""
    n = len(arr)
    for i in range(1, n):
        x = arr[i]
        j = i - 1
        # find where to place x
        while j >= 0 and x < arr[j]:
            arr[j+1] = arr[j]
            j -= 1
        arr[j+1] = x
```

Problem 3.

In each of the parts below, compute the *expected running time* of the given code. Also state the *worst-case* running time. Use asymptotic notation.

For full credit, for each part: (1) clearly state the cases you consider, (2) give the probability of each case, (3) give the time taken in each case, and (4) compute the expected time as a probability-weighted sum.

Reminder. In Python, `np.random.randint(n)` picks a random integer uniformly from $\{0, 1, \dots, n - 1\}$.

```
a) def alpha(n):
    # pick a random integer from 0 to n-1
    r = np.random.randint(n)
    if r < n/5:
        for i in range(n**3):
            print(i)
    else:
        for i in range(n):
            print(i)
```

```
b) def beta(n):
    # pick a random integer from 0 to n-1
    r = np.random.randint(n)
    if r < 20:
        for i in range(n**2):
            print(i)
    else:
        for i in range(n):
            print(i)
```

```
c) def gamma(n):
    # pick a random integer from 0 to n-1
    m = np.random.randint(n)
    total = 0
    for i in range(m):
        for j in range(i):
            total += 1
```

```
d) def random_search(arr, target):
    n = len(arr)
    while True:
        idx = np.random.randint(n)
        if arr[idx] == target:
            return idx
```

Assume all elements in `arr` are distinct and `target` is in the array.