Average Case of Quicksort

Recurrence relations

Set-up

- When it comes to the average case, the position of the pivot does not matter.
- Instead, we consider the sizes of each of the possible subsets.
- We define a cost function T(*n*), which tells the cost of using Quicksort on an array of size *n*.
- Consider an array of size N, then the "left" side of the pivot can be in the range of 0 to N-1 elements.

Set-up

- This gives us an average cost of:
- T(L) = [T(0) + T(l) + T(2) + ... + T(N-1)]/N
- We know that T(L) = T(R)
- The total cost of an iteration of Quicksort consists of two recursive calls and linear time to partition the array around the pivot.
- We obtain:
- T(N) = 2(T(0) + T(l) + T(2) + ... + T(N - 1))/N + (N-1)

 8-3

Set-up towards recurrence

- We multiply both sides by *N*, obtaining:
- NT(N) = 2(T(0) + T(l) + T(2) + ... + T(N - 1)) + N^2 - N
- We then write this equation for the case N - 1, so that we can simplify the equation by subtraction:
- (N-1)T(N-1) = 2(T(0) + T(l) + ... + T(N-2)) + $(N-1)^2$ - (N-1)

 8-4

Recurrence relation

- We subtract the N-1 case from the N case:
- NT(N) - (N-1)T(N-1) = 2T(N-1) + 2N - 2
- We rearrange terms and drop the insignificant -2 on the right-hand side, obtaining:
- NT(N) = (N + 1)T(N-1) + 2N
- We now have a formula for T(N) in terms of T(N-1) only.

Average Case

- The idea is to telescope, but the equation is in the wrong form; while the argument to T goes down, the factor N goes up:

  NT(N) = (N + 1)T(N-1) + 2N
- If we divide the equation by N(N+1), we get:
- T(N)/(N+1) = T(N-1)/N + 2/(N+1)
- We can now substitute the green expression.

Average Case

- Now telescope:
- T(N)/(N+1) = T(N-1)/N + 2/(N+1)
- T(N-1)/N = T(N-2)/(N-1) + 2/N
- T(N-2)/(N-1) = T(N-3)/(N-2) + 2/(N-1)
- …
- T(3)/4 = T(2)/3 + 2/4
- T(2)/3 = T(1)/2 + 2/3
- Notice that T(1) = 0

Average Case

- Adding everything, we obtain:
- T(N)/(N+1) =
  2(1/(N+1) + 1/N + … + 1/4+ 1/3) + 0 =
  2(1/(N+1) + 1/N + … + 1/4+ 1/3 + 1/2 + 1) - 3 =
- Sum of i from 1 to N of 1/i is log(N), by Theorem 5.5
- We get: 2 log(N) - 3
- Multiplying by (N+1) to obtain T(N), gives:
- T(N) = O(N log N)