CSC236 *Intro. to* the Theory of Computation

Lecture 5: Recurrences and D&C

Amir H. Chinaei, Fall 2016

Office Hours: W 5-4 BA4222

ahchinaei@cs.toronto.edu
http://www.cs.toronto.edu/~ahchinaei/

Course page:
http://www.cdf.toronto.edu/~csc236h/fall/index.html

Section page:
http://www.cdf.toronto.edu/~csc236h/fall/amir_lectures.html

review

- so far
  - different variants of induction
  - recurrence relations
  - introduced the application of recurrence relations to complexity of recursive algorithms
- this week
  - application of recurrence relations to complexity of Divide & Conquer algorithms

recursive algorithms

- normally reduce/split the problem to some problems of smaller size
  - *factorial*(*n* - 1) is smaller vs. *factorial*(*n*)
  - *fib*(*n* - 1) and *fib*(*n* - 2) are smaller vs. *fib*(*n*)
  - *mergeSort*(*A* , *1st half*) and *mergeSort*(A, 2nd half) are smaller vs. *mergeSort*(*A*)
  - *binSearch*(x, *A* , *1st half*) and *binSearch*(x, A, 2nd half) are smaller vs. *binSearch*(x, *A*)
- recurrences
  - towards the complexity of D&C Alg.

Example 61: *binSearch*

```
def binSearch(x, A, b, e):
    if b == e:
        if x == A[b]:
            return b
        else:
            return -1
    else:
        m = (b + e) // 2          # midpoint
        if x <= A[m]:
            return binSearch(x, A, b, m)
        else:
            return binSearch(x, A, m+1, e)
```

Example 61: *binSearch*

- a recurrence relation for complexity of *binSearch*

Example 61: *binSearch*

- guessing (roughly calculating) a closed form

Example 61: *binSearch*

- calculating a lower bound

Example 61: *binSearch*

- calculating a lower bound

Example 61: *binSearch*

- calculating an upper bound

Example 61: *binSearch*

- calculating an upper bound

notes:

notes: