CSC236 *Intro. to* the Theory of Computation

Lecture 6: More D&C Complexity

Amir H. Chinaei, Fall 2016

Office Hours: W 2-4 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

- Last week
  - introduced the application of recurrence relations to complexity of d&c algorithms
    - in particular, recursive binary search
- this week
  - application of recurrence relations to complexity of d&c algorithms
    - in particular, merge sort, and closest pairs of points
  - master theorem

Example 63: *mergeSort*

```python
def mergeSort(A,b,e):
    if b == e: return A[b:1]
    m = (b + e) // 2
    mergeSort(A,b,m)
    mergeSort(A,m+1,e)
    # merge sorted A[b..m] & A[m+1..e] back into A[b..e]
    B = A.copy()
    c = b
    d = m+1
    for i in range(b, e+1):
        if d > e or (c <= m and B[c] < B[d]):
            A[i] = B[c]
            c += 1
        else:  # d <= e and (c > m or B[c] >= B[d])
            A[i] = B[d]
            d += 1
    return A
```

Example 63: *mergeSort*

- a recurrence relation for complexity of *mergeSort*

Example 63: *mergeSort* … closed form

$$T(\hat{n}) = \begin{cases} 1 & \hat{n} = 1 \\ 2T\left(\frac{\hat{n}}{2}\right) + \hat{n} + 1 & \hat{n} > 1 \end{cases}$$

$\hat{n} = 2^k$

$=$

$= \hat{n} \log \hat{n} + 2\hat{n}$ -1

Example 63: *mergeSort* ... T(n) increasing

- Since T(n) is increasing (for prove see Lemma 3.6),

$$T\left(\frac{\hat{n}}{2}\right) \leq T(n) \leq T(\hat{n}) \qquad \text{when } 2^{k-1} \leq n \leq 2^k$$

Example 63: *mergeSort*

- calculating a lower bound

Example 63: *mergeSort*

- calculating a lower bound

Example 63: *mergeSort*

- calculating an upper bound