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 pair of points
  - master theorem

Example 63: *mergeSort*

```
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*

$$T(n) = \begin{cases} c_1 & n = 1 \\ c_2 + T(m-b+1) + T(e-m) + n & n > 1 \end{cases}$$

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

Example 63: *mergeSort* ... closed form

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

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

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

$$T(\tfrac{\hat{n}}{2}) \le T(n) \le T(\hat{n}) \qquad \text{when } 2^{k-1} \le n \le 2^k$$

Example 63: *mergeSort*

Keep in mind: $T(\frac{\hat{n}}{2}) \le T(n) \le T(\hat{n})$ and $T(\hat{n}) = \hat{n}\log\hat{n} + 2\hat{n} - 1$

- calculating a lower bound

$T(n) \ge c\,n\log n$

$$
\begin{aligned}
T(n) &\ge T\left(\tfrac{\hat{n}}{2}\right) \\
&= \tfrac{\hat{n}}{2}\log\tfrac{\hat{n}}{2} + 2\tfrac{\hat{n}}{2} - 1 \\
&= \tfrac{\hat{n}}{2}(\log\hat{n} - \log 2) + \hat{n} - 1 \\
&= \tfrac{\hat{n}}{2}\log\hat{n} + \tfrac{\hat{n}}{2} - 1 \\
&\ge \tfrac{n}{2}\log n + \tfrac{n}{2} - 1 \\
&\ge \tfrac{n}{2}\log n
\end{aligned}
$$

$c = \frac{1}{2} \quad n \ge 2$

□

Example 63: *mergeSort*

- calculating a lower bound

Example 63: *mergeSort*

Keep in mind: $T(\frac{\hat{n}}{2}) \le T(n) \le T(\hat{n})$ and $T(\hat{n}) = \hat{n}\log\hat{n} + 2\hat{n} - 1$

- calculating an upper bound

$T(n) \le c\,n\log n$

$$
\begin{aligned}
T(n) &\le T(\hat{n}) \\
&= \hat{n}\log\hat{n} + 2\hat{n} - 1 \\
&\le 2n\log 2n + 2.2n - 1 \\
&= 2n(\log 2 + \log n) + 4n - 1 \\
&= 2n\log n + 6n - 1 \\
&\le 2n\log n + 6n \\
&\le 2n\log n + 6n\log n \\
&\le 8n\log n
\end{aligned}
$$

$c = 8 \quad n \ge 2$

□