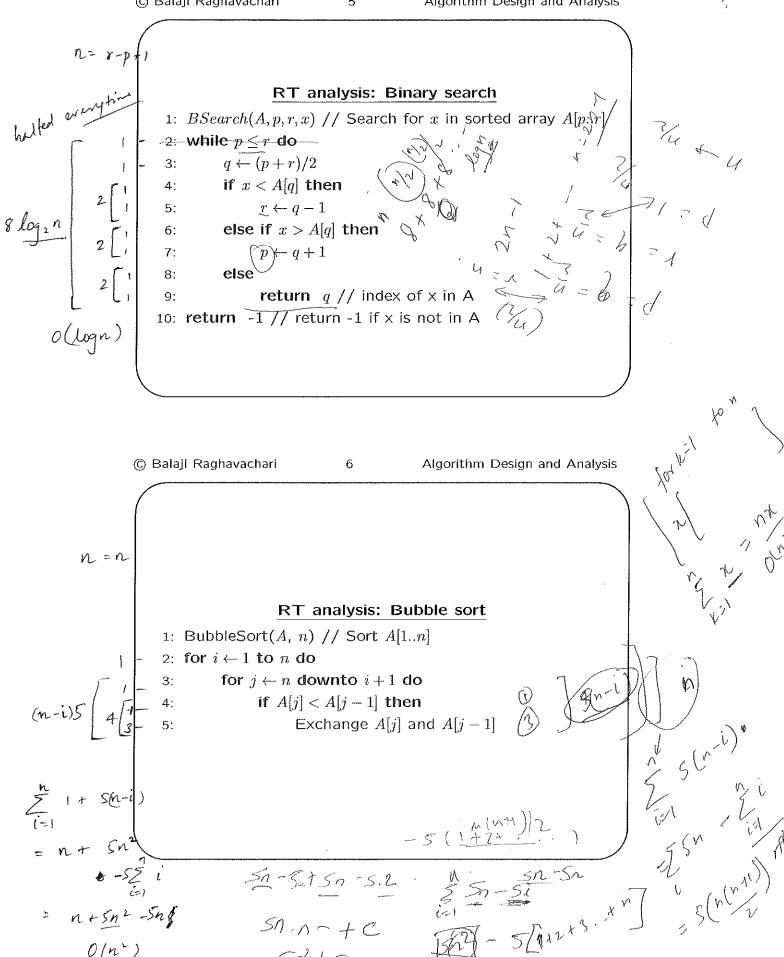
Algorithm Design and Analysis

Balaji Raghavachari 4 Algorithm Design and Analysis n = f(r-p+1) RT analysis: Partition algorithm  $? - 1: i \leftarrow Random(p,r)$  3 - 2: Exchange A[i] and A[r]  $3: x \leftarrow A[r]$   $1 - 4: i \leftarrow p - 1$   $5: \text{ for } j \leftarrow p \text{ to } r - 1 \text{ do}$   $6: \text{ if } A[j] \leq x \text{ then}$   $7: i \leftarrow i + 1$  3 - 8: Exchange A[i] and A[j] 9: Exchange A[i + 1] and A[r] 1 - 10: return i + 1 O(n) ?



## RT analysis: Counting sort

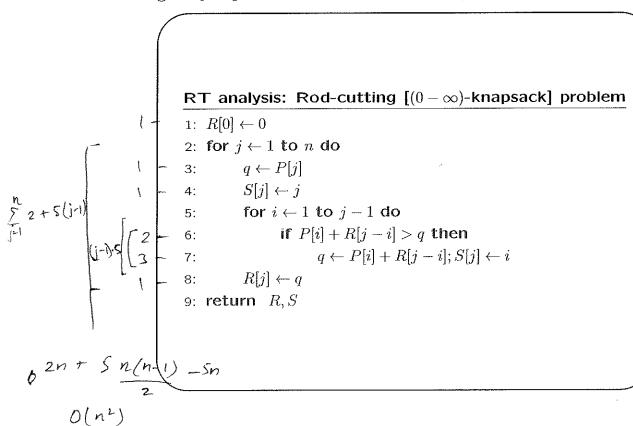
Sort A[1..n] with integer field "key" in range 1..k:

- 1: for  $j \leftarrow 1$  to k do
- 2:  $C[j] \leftarrow 0$
- 3: for  $i \leftarrow 1$  to n do
- 4: C[A[i].key]++
- 5: for  $j \leftarrow 2$  to k do
- 6:  $C[j] \leftarrow C[j] + C[j-1]$
- 7: for  $i \leftarrow n$  downto 1 do
- 7: **for**  $i \leftarrow n$  **downto** 1 **do**2 8:  $j \leftarrow A[i].key$ 2 9:  $B[C[j] -] \leftarrow A[i]$ 
  - 10: return B

© Balaji Raghavachari

8

Algorithm Design and Analysis



MY

```
 \begin{array}{c} \textbf{RT analysis: Activity selection problem} \\ 1: \ ASP[0] \leftarrow 0 \\ 2: \ \textbf{for } i \leftarrow 1 \ \textbf{to } n \ \textbf{do} \\ 3: \quad j \leftarrow i - 1 \\ 4: \quad \textbf{while } f_j > s_i \ \textbf{do} \\ 5: \quad j \leftarrow j - 1 \\ \hline 2: \ \textbf{6:} \quad \textbf{if } ASP[i - 1] \geq ASP[j] + P_i \ \textbf{then} \\ \hline 2: \quad ASP[i] \leftarrow ASP[i - 1]; \quad Use[i] \leftarrow \text{'N'} \\ 8: \quad \textbf{else} \\ 3: \quad ASP[i] \leftarrow ASP[j] + P_i; \quad Use[i] \leftarrow \text{'Y'}; \quad Pre[i] \leftarrow j \\ \hline 10: \ \textbf{return} \quad ASP, Use, Pre \\ \hline \end{array}
```

```
© Balaji Raghavachari
                                    10
                                                  Algorithm Design and Analysis
     RT analysis: Longest common subsequence problem
     1: for i \leftarrow 0 to m do
     2: C[i,0] \leftarrow 0
     3: for j \leftarrow 0 to n do
4: C[0,j] \leftarrow 0
     5: for i \leftarrow 1 to m do
              for j \leftarrow 1 to n do
                     if x[i] = y[j] then
                            C[i,j] \leftarrow C[i-1,j-1] + 1; \quad B[i,j] \leftarrow 'D'
                     else if C[i, j-1] > C[i-1, j] then
                            C[i,j] \leftarrow C[i,j-1]; \quad B[i,j] \leftarrow L'
   10:
                     else
   11.
                            C[i,j] \leftarrow C[i-1,j]; \quad B[i,j] \leftarrow `U'
   12:
   13: return C, B
```

```
RT analysis: Output an actual LCS

1: i \leftarrow m

2: j \leftarrow n

3: k \leftarrow C[m,n]

4: while k > 0 do

5: if B[i,j] = D' then

6: z[k] \leftarrow x[i]

7: k \leftarrow k-1; i \leftarrow i-1; j \leftarrow j-1

8: else if B[i,j] = L' then

29: j \leftarrow j-1

10: else

11: i \leftarrow i-1

12: return z
```

© Balaji Raghavachari

12

Algorithm Design and Analysis

## RT analysis: Huffman coding

Input: Alphabet  $\Sigma$ , frequency of each character. Let  $n = |\Sigma|$ 

1. For each character  $c\in \Sigma$ , create a tree with a single node containing element c, with priority f(c)

n 2. Create a priority queue with these n trees

3. While the priority queue has more than one tree, remove two least fequent trees x and y, merge them together, and insert the resulting tree into the priority queue, with priority f(x) + f(y)

 $\it f$  4. Return the single tree in the priority queue

Under (n) (n)

Onlyn

## RT analysis: APSP by extension

```
1: for u \in V do
```

2: for 
$$v \in V$$
 do

3: 
$$D[u,v] \leftarrow w(u,v)$$

4: 
$$k \leftarrow 1$$

5: while 
$$k < \lvert V \rvert - 1$$
 do

6: for 
$$u \in V$$
 do

7: for 
$$v \in V$$
 do

8: 
$$\int T[u,v] \leftarrow D[u,v]$$

9以代 for 
$$m \in V$$
 do

95 (A) for 
$$m \in V$$
 do
10:  $(V)$   $T[u,v] > D[u,m] + D[m,v]$  then

11: 
$$T[u,v] \leftarrow D[u,m] + D[m,v]$$

12: 
$$D \leftarrow T$$

13: 
$$k \leftarrow 2 * k$$
 -  $(log n)$ 

14: return D

- © Balaji Raghavachari
- 14

Algorithm Design and Analysis

## RT analysis: Floyd-Warshall's algorithm for APSP

1: for  $u \in V$  do

2: for 
$$v \in V$$
 do

3: 
$$D[u,v] \leftarrow w(u,v)$$

4: for 
$$k \leftarrow 1$$
 to  $|V|$  do

5: for 
$$u \in V$$
 do

6: for 
$$v \in V$$
 do

7: 
$$D[u,v] \leftarrow \min(D[u,v], D[u,k] + D[k,v])$$

```
\begin{array}{c} \text{RT analysis: KMP algorithm for string matching} \\ 1: \ \pi[1] \leftarrow 0 \\ 2: \ k \leftarrow 0 \\ 3: \ \text{for } q \leftarrow 2 \ \text{to } m \ \text{do} \\ 3: \ \text{for } q \leftarrow 2 \ \text{to } m \ \text{do} \\ 4: \qquad \text{while } k > 0 \ \text{and } P[k+1] \neq P[q] \ \text{do} \\ 5: \qquad k \leftarrow \pi[k] \\ 6: \qquad \text{if } P[k+1] = P[q] \ \text{then} \\ 7: \qquad k \leftarrow k+1 \\ 8: \qquad \pi[q] \leftarrow k \\ 9: \ \text{return } \pi \\ \\ \hline \\ \mathcal{O}\left(\ \text{km}\right) \end{array}
```

© Balaji Raghavachari

16

Algorithm Design and Analysis

```
## A malysis: Counting sort

for(q=1; q<=k; q++) {
    L[q] = new LinkedList();
    }
    for(i=1; i<=n; i++) {
        L[A[i].key].add(A[i]);
    }
    k = 1;
    for(q=1; q<=k; q++) {
        for(x: L[q]) {
            A[k] = x;
            k++;
        }
     }
}
```