FA HW 5

Xiaoyu Xue

2017-10-15

1 Ex 2.21

1.a

```
    global Hashmap AvailableLetters stores (l, c) pair
    global Array Word[1...n] store letters to print
    procedure DFS(level, k);
1     if level == n+1 then
2       print(Word);
3     endif]
4     foreach i 1 to k do
5       Word[level] ← AvailableLetters[i];
6       AvailableLetters[i].c − −;
7       if AvailableLetters[i].c == 0 then
8         Swap(AvailableLetters[i], AvailableLetters[k]);
9         DFS(level + 1, k − 1);
10      else
11        DFS(level + 1, k);
12      endif
13      AvailableLetters[i].c + +;
14    endfor
    end_DFS;
```

2 Ex 3.29

2.a

$A(n) = \Theta(n \log n)$;

2.b

$B(n) = \Theta(n \log n)$;

2.c

$C(n) = \Theta(n)$;

2.d

$D(n) = \Theta(n)$;

2.e

$E(n) = \Theta(n^2)$;

2.f

$F(n) = \Theta(n^2)$;

3 Ex 3.30

3.a

$$\begin{cases} \text{T(1)} = 1 & if\ n\ =\ 1; \\ \text{T(n)} = 1+2\text{T(n-1)}; & if\ n\ >\ 1; \end{cases}$$

3.b

$$\begin{cases} \text{U(1)} = 1 & if\ n\ =\ 1; \\ \text{U(n)} = \text{T(n-1)}+2\text{U(n-1)}; & if\ n\ >\ 1; \end{cases}$$

3.c

$$\begin{cases} \text{W(1)} = 1 & if\ n\ =\ 1; \\ \text{W(n)} = 3\text{W(n-1)}; & if\ n\ >\ 1; \end{cases}$$

4 Ex 4.1

4.a

Given two sequences A and B, find subsequence with the largest count k for a that with indices $i1 < i2 < i3 < ... < ik$ and $j1 < j2 < j3 < ... < jk$ where $A[i1] = B[j1]$, $A[i2] = B[j2]$... and so on.

4.b

$$\text{S(i, j)} = \begin{cases} 0, & if\ i\ =\ 0\ or\ j\ =\ 0 \\ \text{S(i-1, j-1)}+1, & if\ A[i]\ =\ B[j]; \\ \text{max(S(i-1, j), S(i, j-1))}, & if\ A[i]\ !=\ B[j]; \end{cases}$$

5 Ex 4.2

5.a

Given a piece of wood and an value data array $Val[i, j]$ for $0 <= i < j < n$;
find a count l and indices $i1$ to il that maximizes the value of $Val[0, i1] + Val[i1, i2] + ... + Val[il, n]$.

5.b

S(i) = $\begin{cases} 0 & if\ i\ =\ 0; \\ \max(Val[k,i]+S(k)); \text{ for all k that } 0 =< k < i & if\ i\ >\ 0; \end{cases}$

6 4.3

6.a

```
    global m, n A[1..n], B[1..m];
    function Length(i, j);
1     if i == 0 or j == 0 then
2       return(0);
3     endif]
4     if A[i] == B[j] then
5       temp = Length(i-1, j-1) +1;
6     else
7       temp = max(Length(i-1, j), Length(i, j-1));
8     endif
9     return(temp);
    end_Length;
```

6.b

```
    global m, n A[1..n], B[1..m];
    global Look[1..n, 1..m] lookup table store subproblems's result with initial value -1;
    function Length(i, j);
1     if i == 0 or j == 0 then
2       return(0);
3     endif]
4     temp ← Look[i, j];
5     if temp > −1 then
6       return(temp);
7     endif
8     if A[i] == B[j] then
9       temp ← Length(i − 1, j − 1) + 1;
10    else
11      temp ← max(Length(i − 1, j), Length(i, j − 1));
```

```
12      endif
13      Look[i, j] ← temp;
14      return(temp);
    end_Length;
```

7 Ex 4.4

```
    global m, n A[1..n], B[1..m];
    global Look[1..n, 1..m] lookup table store subproblems's result with initial value -1;
    global whereto[1..n, 1..m] store subproblems's solution;
    function Length(i, j);
 1    if i == 0 Or j == 0 then
 2      return(0);
 3    endif]
 4    temp ← Look[i, j];
 5    if temp > −1 then
 6      return(temp);
 7    endif
 8    if A[i] == B[j] then
 9      temp ← Length(i − 1, j − 1) + 1;
10      (r, s) ← (i − 1, j − 1);
11    elseif Length(i − 1, j) > Length(i, j − 1) then
12      temp ← Length(i − 1, j);
13      (r, s) ← (i − 1, j);
14    else
15      temp ← Length(i, j − 1);
16      (r, s) ← (i, j − 1);
17    endif
18    whereto[i, j] ← (r, s);
19    Look[i, j] ← temp;
20    return(temp);
    end_Length;

    procedure PrintPath(i, j);
 1    if i > 0 And j > 0 then
 2      (r, s) ← whereto[i, j];
 3      PrintPath(r, s);
 4      if r == i − 1 And s == j − 1then
 5        print(A[i]);
 6      endif
 7    endif]
    end_PrintPath;
```