600.363/463 Algorithms - Fall 2013
Solution to Assignment 1

(110 points)

I (30 points) Tow possible solutions are shown below.

Solution 1: i. Algorithm(10 points)

Input: Two arrays A and B, both of length n

Output: True if A and B have at least one common element. False otherwise.

```
1  flag = False;
2  for i ← 1 to n do
3  |   key ← A[i];
4  |   for j = 1 to n do
5  |   |   if B[j]==key then
6  |   |   |   flag← True;
7  |   |   |   break;
8  |   |   end
9  |   end
10 end
11 return flag;
```

ii. Correctness(10 points)

Loop invariant: At the start of each iteration of the for loop in of lines 2-10, if flag=False, the subarrays $A[1..i-1]$ and $B[1..n]$ do not have any common element; if flag=True, the subarrays $A[1..i-1]$ and $B[1..n]$ do not have any common element and $A[i] \in B[1..n]$.

Initialization (input to loop): $i = 1$, $A[1..i-1]$ is empty, nothing need to be proved.

Maintenance (loop to loop): new $i = i_{old} + 1$, flag=False, $A[i-1] \notin B[1..n]$. Hence loop invariant holds.

Termination (loop to out): If flag=True, for current i, $A[i] \in B[1..n]$ and $A[i-1] \notin B[1..n]$; If flag=False, $i = n+1$, $A[n] \notin B[1..n]$. Then we conclude that $A[1..n]$ does not have common element with $B[1..n]$, Hence the algorithm is correct.

iii. Speed(10 points)

i takes less than or equal to n values and j takes less than or equal to n values. It takes $O(1)$ for each comparison, hence $T(n) = O(n^2)$.

Solution 2: i. Algorithm

```
Input: Two arrays A and B, both of length n
Output: True if A and B have at least one comment element. False otherwise.
1  Sort A and B using Merge-Sort;
2  i ← 1;
3  j ← 1;
4  flag ← False;
5  while i <= n and j <= n do
6  |  if A[i] == B[j] then
7  |  |  flag←True;
8  |  |  break;
9  |  end
10 |  else if A[i] < B[j] then
11 |  |  i++;
12 |  end
13 |  else
14 |  |  j++;
15 |  end
16 end
17 return flag;
```

ii. Correctness

Correctness of merge sort refers to textbook.

Loop invariant: At the start of each iteration of the while loop in of lines 5-16, if flag=False, the subarrays $A[1..i-1]$ and $B[1..j-1]$ do not have any common element; if flag=True, the subarrays $A[1..i-1]$ and $B[1..j-1]$ do not have any common element and $A[i] = B[j]$.

Initialization (input to loop): $i = 1, j = 1$, $A[1..i-1]$ and $B[1..j-1]$ are empty, nothing need to be proved.

Maintenance (loop to loop): flag must be False, if new $i = i_{old} + 1$, $A[i-1] \notin B[1..j-1]$; if new $j = j_{old} + 1$, $B[j-1] \notin A[1..i-1]$;. Hence loop invariant holds.

Termination (loop to out): If flag=True, for current i, j, $A[i] = B[j]$, but $A[i-1] \notin B[1..j-1]$ and $B[j-1] \notin A[1..i-1]$; If flag=False, then if $i = n+1$, $A[n] \notin B[1..j-1]$ and if $j = n+1$, $B[n] \notin A[1..i-1]$. Then we conclude that $A[1..n]$ does not have common element with $B[1..n]$, Hence the algorithm is correct.

iii. Speed

Merge-sort takes $O(n \log n)$ time. i, j takes less than or equal to n values respectively, and each comparison takes $O(1)$ time, hence $T(n) = O(n \log n) + O(n) = O(n \log n)$

II (10 points) For $n > 1$,

$$\begin{aligned}f(n) =& 2f(n-1) + n \\ =& 2(2f(n-2) + n - 1) + n \\ =& 2^2 f(n-2) + 2(n-1) + n \\ =& \cdots \\ =& 2^{n-1} f(1) + 2^{n-2}(n - (n-2)) + \cdots + 2(n-1) + n \\ =& 1 \cdot 2^{n-1} + 2 \cdot 2^{n-2} + 3 \cdot 2^{n-3} + \cdots + (n-1) \cdot 2^1 + n \cdot 2^0\end{aligned}$$

Then

$$2f(n) = 1 \cdot 2^n + 2 \cdot 2^{n-1} + 3 \cdot 2^{n-2} + 4 \cdot 2^{n-3} + \cdots + n \cdot 2^1$$

By subtracting the above two equations, we have

$$f(n) = 2^n + 2^{n-1} + \cdots + 2 + 1 - n - 1 = 2^{n+1} - n - 2 \tag{1}$$

To prove by induction, when $n = 1$,

$$f(1) = 2^2 - 1 - 2 = 1$$

Assume for $n = k, k = 1, 2, 3, \cdots$, $f(k) = 2^{k+1} - k - 2$ holds, then for $n = k + 1$,

$$f(k+1) = 2f(k) + (k+1) = 2(2^{k+1} - k - 2) + k + 1 = 2^{k+2} - (k+1) - 2$$

Hence the claim holds.

III (70 points) Note that here we assume log means $\log_2$.

1 (10 points) T.
$3n^2 + 6n \le 9n^2$ for any $n \ge 3$, hence $3n^2 + 6n = O(n^2)$.

2 (10 points) T.
$3n^2 + 6n \le 6n^2 \log n$ for any $n \ge 2$,hence $3n^2 + 6n = O(n^2 \log n)$.

3 (10 points) F.
$O(\log n) > O(1)$. More precisely, given any n_0 and c, $n^2 \log n > cn$ when $n \ge \max\{2^c, n_0\}$.

4 (10 points) F.
$3^n = O(\frac{3}{2})^n O(2^n) > O(2^n)$. More precisely, given any n_0 and c, $3^n > c * 2^n$ when $n \ge \max\{\log_{3/2} c, n_0\}$.

5 (10 points) F.
Note that taking log preserves inequality. If $\log n \le c(\log\log n)^4$, then $\log\log n \le \log c + 4\log\log\log n$. Clearly $\log\log n \ge \log\log\log n$, hence it is false.

6 (10 points) T
Note that taking log preserves inequality. If $n \le c(\log n)^{\log n}$, then $\log n = \log c + \log n(\log\log n)$. Clearly this holds.

7 (10 points) T
Note that taking log preserves inequality. If $n^{100} \le c2^n$, then $\log n^{100} = 100\log n \le \log c + \log 2^n = \log c + n\log 2$. Clearly this holds.