1- به اسلایدها رجوع کنید.

2- الگوریتم بازگشتی:

```
Print_Station(L, i, j)
    if j > 1
        Print_Station(L, L[i,j], j-1)
    Print "line" i ", station " j ". "
```

طبیعتا فراخوانی تابع فوق (در main) هم به صورت Print_Station(L, l*, n) است.

3- بله. در هر تکرار الگوریتم نیازی به نگهداری تمام آرایه f نیست و می‌توان تنها دوعنصر مربوط به ایستگاه iام و i-1ام را نگه داشته و بقیه را دور ریخت.

5- بله چون در این حالت هم ساختار بهینه قابل تشکیل است مشابه روش پویا برای حالت مینیمم می‌توان حالت ماکزیمم را هم بدست آورد با این تفاوت که بجای min در رابطه بازگشتی از max استفاده می‌کنیم.

7- برای محاسبه طول بزرگترین زیررشته مشترک وقتی به صورت سطری پیش می‌رویم فقط به عناصر سطر قبل احتیاج داریم که اگر رشته کوتاهتر را در بالای ماتریس قرار داده باشیم، فضای لازم را از $m \times n$ به $2 \times \min(m, n)$ تقلیل می‌دهد. این فضا باز هم قابل کاهش است به این صورت که برای محاسبه هر عنصر فقط عناصر قبل از عنصر جاری از سطر قبل را ممکن است نیاز داشته باشیم و بقیه قابل دور ریختن هستند پس می‌توانیم فقط به اندازه یک سطر عنصر را نگه داریم و هعنشری که دیگر نیاز نیست با مقدار جاری جایگزین شود که در این صورت به مقدار $\min(m, n) + 1$ حافظه احتیاج داریم.

8- زمان اجرای الگوریتم پیشنهادشده با DP برای محاسبه LCS برابر با $O(n^2)$ است. پس اگر رشته دومی بسازیم که حاوی یک دنباله مرتب صعودی (نزولی) از محدوده اعداد موجود در رشته باشد و زیردنباله مشترک آنها را به روش فوق بسازیم، یک دنباله اکیدا یکنواخت صعودی (نزولی) ساخته‌ایم در زمان $O(n^2)$.