تحلیل و طراحی الگوریتم‌ها — سری اول تمرین‌ها

۱) توابع زیر را با توجه به رشد آنها مرتب نمایید:

$$f_۱ = n^{۲/۵}\text{، } f_۲ = \sqrt{۲n}\text{، } f_۳ = n + ۱۰\text{، } f_۴ = ۱۰^n\text{، } f_۵ = ۱۰۰^n\text{، } f_۶ = n^۲ \log n.$$

۲) نشان دهید $n(\log_۳ n)^۵ = O(n^{۳/۲})$.

۳) رابطه‌ی بازگشتی زیر را با فرض $T(۱) = ۱$ و $T(۲) = ۴$ حل کنید.

$$T(n) = ۸\,T(n-۱) + ۱۵\,T(n-۲)$$

۴) الگوریتمی با پیچیدگی زمانی $O(n^۲)$ برای مسئله‌ی «سه–مجموعه» ارائه دهید. راهنمایی: در حلقه‌ی دوم الگوریتمی که در کلاس ارائه شد، مجموع $A[i] + A[j]$ افزایش می‌یابد و در نتیجه $-(A[i] + A[j])$ کاهش می‌یابد؛ سعی کنید همزمان با افزایش j، مقدار k را کاهش دهید.

۵) الگوریتمی با پیچیدگی زمانی $O(n \log n)$ برای مسئله‌ی «بزرگ‌ترین زیر رشته‌ی متوالی» ارائه دهید. راهنمایی: با ذخیره‌سازی بزرگ‌ترین رشته‌ی متوالی شروع شده از اعداد بررسی شده در A، سعی کنید حلقه‌ی دوم الگوریتم را جایگزین کنید.

یادآوری مسئله‌ی سه–مجموع

صورت مسئله: الگوریتمی ارائه دهید که با گرفتن مجموعه‌ای از اعداد اعشاری مثل $A = \{a_۱, a_۲, \ldots, a_n\}$، حداقل یکی از زیر مجموعه‌های سه عضوی آن که مجموع عناصر آن صفر باشد را بیابد. برای نمونه در مجموعه‌ی $A = \{۱, -۱, ۴, -۳, ۲\}$، زیر مجموعه‌ی $\{۱, ۲, -۳\}$ یکی از جواب‌های ممکن است.

توضیحات: در کلاس الگوریتم‌هایی با پیچیدگی زمانی بدترین حالت $O(n^۳)$ (سه حلقه‌ی تو در تو) و $O(n^۲ \log n)$ (با مرتب کردن اعداد و استفاده از جستجوی دودویی) ارائه شد.

یادآوری مسئله‌ی بزرگ‌ترین زیر رشته‌ی متوالی

صورت مسئله: دنباله‌ی $A = (a_1, a_2, \ldots, a_n)$ از اعداد صحیح را در نظر بگیرید. الگوریتمی ارائه دهید که طول بزرگ‌ترین زیر رشته‌ی متوالی از دنباله‌ی A را بیابد. یک زیر رشته‌ی متوالی از A رشته‌ای مثل $a_{i_1} a_{i_2} \cdots a_{i_m}$ است که در آن $i_1 < i_2 < \cdots < i_m$ است و $a_{i+1} = a_i + 1$ به ازای $1 \leq i < n$ باشد.

الگوریتم: الگوریتم ساده‌ی زیر با پیچیدگی زمانی $O(n^2)$ این مقدار را محاسبه می‌کند.

```
# A: the input sequence
# n: the number of items in A
algorithm longestsubseq(A, n)
      best_m := -1;
      for i = n downto 1
            m := 0;
            for j = i to n
                  if A[j] == A[i] + m
                        m := m + 1;
            if m > best_m
                  m := best_m;
      return best_m;
```

اثبات درستی الگوریتم: فرض کنید الگوریتم ارائه شده طول بهترین جواب را بر نگرداند. در این صورت زیر رشته‌ای مثل $r = a_{i_1} a_{i_2} \cdots a_{i_m}$ وجود خواهد داشت که طول آن از مقداری که الگوریتم بالا بر می‌گرداند بزرگ‌تر است. در ادامه رشته‌ای را می‌سازیم که طول آن برابر m است و الگوریتم بالا آن را بررسی می‌کند. پس این فرض باطل خواهد بود و الگوریتم به درستی مقدار m را بر می‌گرداند.

فرض کنیم رشته‌ی $s = a_{j_1} a_{j_2} \cdots a_{j_{m'}}$ رشته‌ای باشد که در دور $i = i_1$ از حلقه‌ی بیرونی الگوریتم بررسی می‌شود. بدیهی است که $i_1 = j_1$. فرض می‌کنیم k آخرین اندیسی از i و j باشد که i_k برابر j_k باشد؛ یعنی $i_l = j_l$ به ازای مقادیر l در $1 \leq l \leq k$. همچنین فرض کنیم از بین جواب‌های مسئله با طول m، رشته‌ی r به صورتی انتخاب گردد که مقدار k بیشینه باشد. حال نشان می‌دهیم $m = m'$. در غیر این صورت خواهیم داشت $m > m'$. دو حالت با توجه به مقدار k نسبت به m' رخ خواهد داد:

در صورتی که $k = m'$ باشد، s زیر رشته‌ای از r خواهد بود. چون مقدار i_{k+1} از j_k بزرگ‌تر است

(چون رشته‌ی r یک جواب مسئله است) و چون مقدار $a_{i_{k+1}}$ برابر $a_{i_k}+1$ است در حلقه‌ی داخلی الگوریتم ارائه شده حتما پس از a_{j_k} عدد $a_{i_{k+1}}$ نیز انتخاب می‌شد. در نتیجه این حالت هیچ گاه رخ نمی‌دهد و $m' > k$ برقرار است.

در صورتی که $k < m'$ باشد، نشان می‌دهیم رشته‌ی $r' = a_{i_1} \cdots a_{i_k} a_{j_{k+1}} a_{i_{k+2}} \cdots a_{i_m}$ نیز یک جواب از مسئله می‌باشد (تنها تفاوت رشته‌ی r' با r جایگزینی عنصر $a_{i_{k+1}}$ با عنصر $a_{j_{k+1}}$ می‌باشد). چون مقدار $a_{i_k} = a_{j_{k-1}} + 1$ و $a_{j_{k+1}} = a_{i_k} + 1$، توالی عناصر در r' حفظ می‌شود. از طرفی، چون i_{k+1} بزرگ‌تر $i_k = j_k$ است و چون $i_{k+2} > i_{k+1} > j_{k+1}$ (با توجه به حلقه‌ی داخلی الگوریتم، کوچکترین اندیس ممکن بزرگ‌تر از i_k انتخاب می‌شود)، خواهیم داشت $i_{k+2} > j_{k+1} > i_k$ و ترتیب اندیس‌ها نیز در r' حفظ می‌شود. پس r' نیز یک زیر رشته‌ی متوالی از A است. در این صورت r' انتخاب r را با شرط حداکثر بودن مقدار k در بین جواب‌های مسئله نقض می‌کند. بنابراین فرض خلف باطل است.