جواب تمرین سری اول

طراحی الگوریتم
دانشکده ریاضی. دانشگاه صنعتی خواجه نصیرالدین طوسی. پاییز ۱۴۰۱

۱. خط چهارم در قطعه کد زیر چند بار اجرا می‌شود؟

```
1. i = 1
2. while(i < n):
3.     i = i * 3
4.     count = count + 1
```

$\lceil \log_3 n \rceil$

۲. در قطعه کد زیر، خطهای سوم و پنجم هر کدام چند بار اجرا می‌شوند؟

```
1. for i in range(1,n):
2.   while (j < n) :
3.     j = j * 2
4.     for k in range(1,j):
5.       count = count + 1
```

بستگی به مقدار اولیه j دارد. با فرض اینکه مقدار اولیه j برابر با 1 باشد، خط سوم به تعداد $\lceil \log_2 n \rceil$ بار در کل اجرا می‌شود. مقدار j که به n رسید، دیگر حلقه while اجرا نمی‌شود. خط پنجم به تعداد

$$2 + 4 + 8 + \ldots + 2^r$$

اجرا می‌شود وقتی که $r = \lceil \log_2 n \rceil$.

۳. در الگوریتم مرتب سازی انتخابی selection sort هر بار کوچکترین عضو لیست مرتب نشده پیدا می‌شود و در انتهای لیست مرتب شده قرار می‌گیرد.

پیاده‌سازی این الگوریتم با زبان پایتون در قطعه کد زیر آمده است.

```
def selectionSort(array, size):

    for ind in range(size):
        min_index = ind

        for j in range(ind + 1, size):
            # select the minimum element in every iteration
            if array[j] < array[min_index]:
                min_index = j
         # swapping the elements to sort the array
        (array[ind], array[min_index]) = (array[min_index], array[ind])
```

برای هر یک از دنباله‌های زیر تعداد مقایسه‌ای که الگوریتم مرتب سازی انتخابی و حبابی انجام می‌دهد را با استفاده از نماد $\Theta()$ و پارامترهای n و k بنویسید. دقت کنید هر دنباله جایگشتی از $1, 2, \ldots, n$ است.

آ) $2, 3, 4, \ldots, n, 1$

- حبابی $\Theta(n^2)$
- انتخابی $\Theta(n^2)$

ب) $n, 1, 2, 3, 4, \ldots, n-1$

- حبابی $\Theta(n)$
- انتخابی $\Theta(n^2)$

ج) $n, n-1, n-2, \ldots, 3, 2, 1$

- حبابی $\Theta(n^2)$
- انتخابی $\Theta(n^2)$

د) $1, 2, 3, \ldots, k, n, n-1, n-2, \ldots, k+1$

- حبابی $\Theta(n(n-k))$
- انتخابی $\Theta(n^2)$

۴. توابع زیر را بر اساس کران مجانبی $\Theta()$ مرتب کنید.

$$f_1(n) = n^{2.5} \qquad f_2(n) = 2n + 3 \qquad f_3(n) = n^{\log n}$$
$$f_4(n) = 2^n \qquad f_5(n) = \log(n!) \qquad f_6(n) = (\log n)!$$

- $2n + 3 \in \Theta(n)$
- $\log(n!) \in \Theta(n \log n)$
- $n^{2.5} \in \Theta(n^{2.5})$
- $(\log n)! \in \Theta(n^{\log \log n})$
- $n^{\log n} \in \Theta(n^{\log n}) = \Theta(2^{(\log^2 n)})$
- $2^n \in \Theta(2^n)$

۵. برای هر کدام از حالات زیر یک تابع مثال بزنید.

آ) $f(n+1) \in O(f(n))$

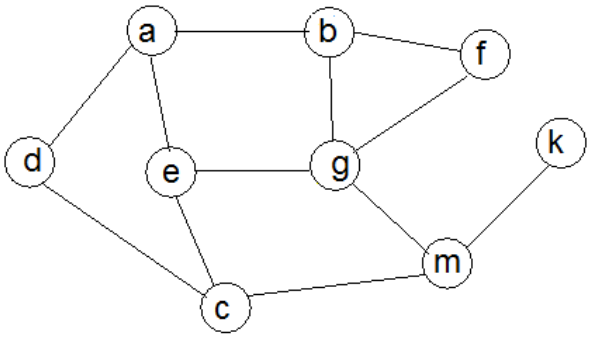
$f(n) = n$

(ب) $f(n+1) \notin O(f(n))$
$f(n) = n!$

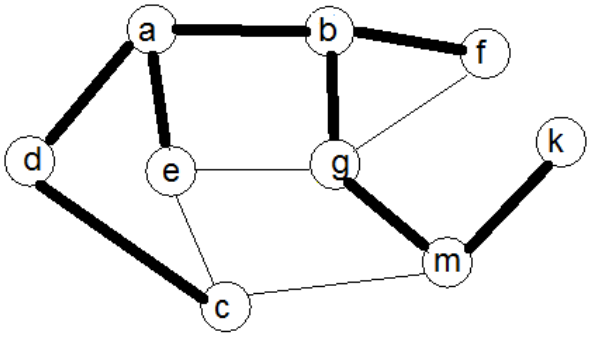
(ج) $f(2n) \notin O(f(n))$ ولی $f(n+1) \in O(f(n))$
$f(n) = 2^n$

(د) $f(2^n) \in O(f(n))$
$f(n) = \log^* n$ $\log^* n$ برابر با تعداد دفعاتی است که عملگر $\log_2$ روی n اعمال شود تا اینکه به کمتر از یا مساوی 1 برسیم.

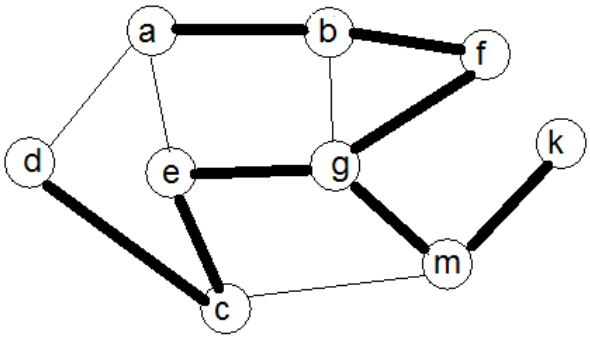
۶. گراف زیر داده شده است. درختهای BFS و DFS را با شروع از راس a بدست آورید. (برای انتخاب همسایه بعدی ترتیب الفبا را در نظر بگیرید.)



BFS tree



DFS tree



۷. چگونه می‌توانیم از الگوریتم DFS برای تشخیص دور در گراف استفاده کنیم؟ زمان اجرای الگوریتم شما چقدر است؟

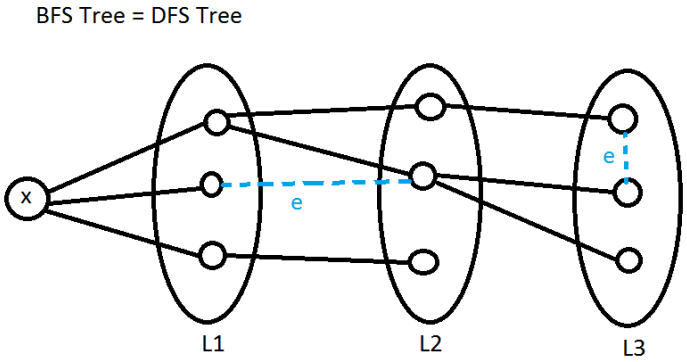
فرض کنید گراف ورودی $G = (V, E)$ باشد. در راه حل زیر فرض کرده ایم که G همبند است. اگر G همبند نباشد برای هر مولفه همبند G راه حل ارائه شده را اجرا میکنیم.

ایده اول: الگوریتم DFS را از یک راس دلخواه $s \in V$ اجرا میکنیم و درخت DFS را تولید میکنیم. فرض کنید $T = (V, E')$ درخت تولید شده باشد. اگر $E' = E$ آنگاه گراف ورودی درخت است و دور ندارد، در غیر اینصورت دور دارد. زمان اجرا $(n + m)$ است.

ایده دوم: هنگام اجرای الگوریتم DFS، یالی که به راس قبلا ملاقات شده وصل باشد، علامت دور در گراف است. به محض برخورد به چنین یالی الگوریتم را متوقف کرده و گزارش میکنیم که گراف دور دارد. زمان اجرا $O(n)$ است چون تعداد یالی که ملاقات میشود تا زمانیکه الگوریتم متوقف شود حداکثر $n + 1$ است.

۸. فرض کنید $G = (V, E)$ یک گراف همبند و ساده (غیر جهت دار) باشد. نشان دهید اگر درخت BFS و DFS با شروع از راس x یکسان باشند آنگاه G حتما یک درخت است.

اثبات با برهان خلف. فرض کنید G درخت نیست. در عین حال درخت BFS و DFS با شروع از راس دلخواه x یکسان هستند.



شکل بالا درخت BFS را نشان میدهد. لایه L_i رئوسی است که فاصله i از راس مبدا x دارند. اگر گراف ورودی درخت نباشد پس باید یال e باشد که در درخت BFS نیست. یال e حتما دور ایجاد میکند. دقت کنید فقط دو حالت میتواند وجود داشته باشد. حالت اول: یال e درون یک لایه واقع شده است. حالت دوم: یال e بین دو لایه متوالی قرار گرفته است. مانند شکل بالا.

از طرف دیگر وجود چنین یالی مانند e با درخت DFS متناقض است. چون پیمایش DFS باید یال e را در پیمایش خود انتخاب کرده باشد.