سوال ۱

برای برگزاری مسابقه ای، تعداد n شرکت کننده در کنار یکدیگر ایستاده اند. نیمی از این شرکت کنندگان از گروه A و نیمی دیگر از گروه B هستند. به هر نفر شماره ای اختصاص داده شده که برای تمایز بین گروه ها، شماره افراد گروه B به صورت منفی نمایش داده می شود. پیش از شروع مسابقه، شرکت کنندگان باید طبق قوانین و دستورات داوران مرتب شوند.

- قانون ۱: افراد گروه A و B باید به صورت یک در میان کنار هم قرار گیرند.
- قانون ۲: اولین نفر باید از گروه A باشد.
- قانون ۳: ترتیب فعلی افراد در هر گروه نباید تغییر کند.

الگوریتمی برای حل این مساله ارایه دهید و آن را به صورت شبه کد بنویسید و از نظر هزینه زمانی بررسی کنید.

مثال:

ورودی : $[-3, -9, 2, 1, 4, -1]$
خروجی : $[2, -3, 1, -9, 4, -1]$

پاسخ:
با توجه به اینکه آرایه قابلیت دسترسی سریع به هر اندیس را دارد از ساختمان داده آرایه استفاده می کنیم.
الگوریتم : یک آرایه جدید تعریف می کنیم. گروه A را عدد مثبت در نظر می گیریم و گروه B را اعداد منفی در نظر میگیریم.
از اولین اندیس شروع می کنیم. اگر عدد مثبت بود در اندیس ۰ آرایه درج می شود و اگر منفی بود در اندیس ۱ آرایه درج می شود و به همین ترتیب ادامه می دهیم به صورت کلی اعداد مثبت اندیس های زوج قرار میگیرند (۰و ۲و ۴و ..) و اعداد منفی در اندیس های فرد قرار میگیرند (۱و ۳و ۵و ..)
پیچیدگی زمانی : $\mathcal{O}(n)$

```
func(a):
        result = []
        j=0
        k=0
        for i=0 to n:
                if a[i] >= 0:
                        result[2j] = a[i]
                        j++
                else:
                        result[2k+1] = a[i]
                        k++
        return result
```

سوال ۲

روابط بازگشتی زیر را حل کنید.

(۱) $T(n) = T(\frac{n}{4}) + T(\frac{3n}{5}) + n^2$

(۲) $T(n) = \sqrt{n}T(\sqrt{n}) + 2n$

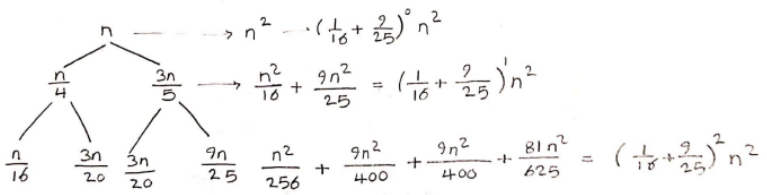
(۳) $T(n) = 4T(\frac{n}{2}) + \frac{n^2}{\log n}$

(۴) $T(n) = T(\sqrt{n}) + \mathcal{O}(\log(\log n))$

پاسخ:

بخش ۱

$$T(n) = T(\frac{n}{4}) + T(\frac{3n}{5}) + n^2$$



$$T(n) = \sum_{i=0}^{\log_{\frac{5}{3}} n} n^2(\frac{1}{16} + \frac{9}{25})^i = n^2 \sum_{i=0}^{\log_{\frac{5}{3}} n} (\frac{34}{400})^i$$

$$T(n) = cn^2 \implies T(n) \in \mathcal{O}(n^2)$$

بخش ۲

$$T(n) = \sqrt{n}T(\sqrt{n}) + 2n$$

$$T(n) = n^{\frac{1}{2}}T(n^{\frac{1}{2}}) + 2n$$

$$2^a = n, \quad T(2^a) = 2^{\frac{a}{2}}T(2^{\frac{a}{2}}) + 2^{a+1}$$

$$S(a) = 2^{\frac{a}{2}}S(\frac{a}{2}) + 2^{a+1}$$

$$S(\frac{a}{2}) = 2^{\frac{a}{4}}S(\frac{a}{4}) + 2^{\frac{a}{2}+1}, \quad S(\frac{a}{4}) = 2^{\frac{a}{8}}S(\frac{a}{8}) + 2^{\frac{a}{4}+1}$$

$$S(a) = 2^{\frac{a}{2}+\frac{a}{4}+\frac{a}{8}} T(\frac{a}{8}) + 2^{\frac{a}{2}+\frac{a}{4}+\frac{a}{4}+1} + 2^{\frac{a}{2}+\frac{a}{2}+1} + 2^{a+1}$$

$$S(a) = 2^{\sum_{j=1}^{k} \frac{a}{2^j}} S(\frac{a}{2^k}) + \sum_{j=1}^{k} 2^{a+1}$$

$$\frac{a}{2^k} = 1 \implies k = \log a, \quad S(a) = 2^{a \sum_{j=1}^{\log a} \frac{1}{2^j}} S(1) + 2^{a+1} \sum_{j=1}^{\log a} 1$$

$$S(a) = 2^a S(1) + \log a * 2^{a+1}$$

$$S(a) \in \mathcal{O}(\log a * 2^{a+1}) \implies T(n) \in \mathcal{O}(n \log\log n)$$

بخش ۳

$$T(n) = 4T(\frac{n}{2}) + \frac{n^2}{\log n}$$

$$T(\frac{n}{2}) = 4T(\frac{n}{4}) + \frac{(\frac{n}{2})^2}{\log \frac{n}{2}} \quad \text{and} \quad T(\frac{n}{4}) = 4T(\frac{n}{8}) + \frac{(\frac{n}{4})^2}{\log \frac{n}{4}}$$

$$T(n) = 4^3 T(\frac{n}{8}) + \frac{16(\frac{n}{4})^2}{\log \frac{n}{4}} + \frac{4(\frac{n}{2})^2}{\log \frac{n}{2}} + \frac{n^2}{\log n}$$

$$T(n) = 4^k T(\frac{n}{2^k}) + \sum_{j=0}^{k-1} \frac{n^2}{\log n - j}$$

$$\frac{n}{2^k} = 1 \implies k = \log n$$

$$T(n) = 4^{\log n} T(1) + \sum_{j=0}^{\log n - 1} \frac{n^2}{\log n - j}$$

$$T(n) = n^2 T(1) + n^2 \sum_{j=0}^{\log n - 1} \frac{1}{\log n - j} \implies \mathcal{O}(n^2 \log\log n)$$

بخش ۴

$$T(n) = T(\sqrt{n}) + \mathcal{O}(\log\log n)$$

$$T(n) = T(\sqrt{n}) + \log\log n$$

$$n = 2^a \quad \text{and} \quad T(2^a) = T(2^{\frac{a}{2}}) + \log\log 2^a$$

$$S(a) = S(\frac{a}{2}) + \log a$$

$$\sum_{i=0}^{\log a} \log \frac{a}{2^i} = \sum_{i=0}^{\log a} \log a - \sum_{i=0}^{\log a} i = (\log a)^2 - \frac{1}{2}(\log a)(\log a + 1)$$

$$T(n) \in \mathcal{O}((\log a)^2)$$

سوال ۳

مرتبه زمانی شبه کد زیر را مشخص کنید.

```
for (i=n/2 ; i<=n ; i++);
        for (j=i ; j<=n ; j++);
                for (k=1 ; k<=n ; k=k*2);
                        count++ ;
```

پاسخ:

$$\sum_{i=\frac{n}{2}}^{n}\sum_{j=i}^{n}\sum_{k=1,2,4,\dots,n}1 \implies \sum_{i=\frac{n}{2}}^{n}\sum_{j=i}^{n}\sum_{k=1}^{\log n}1$$

$$\sum_{i=\frac{n}{2}}^{n}\sum_{j=i}^{n}\log n \implies \sum_{i=\frac{n}{2}}^{n}(n-i+1)\log n$$

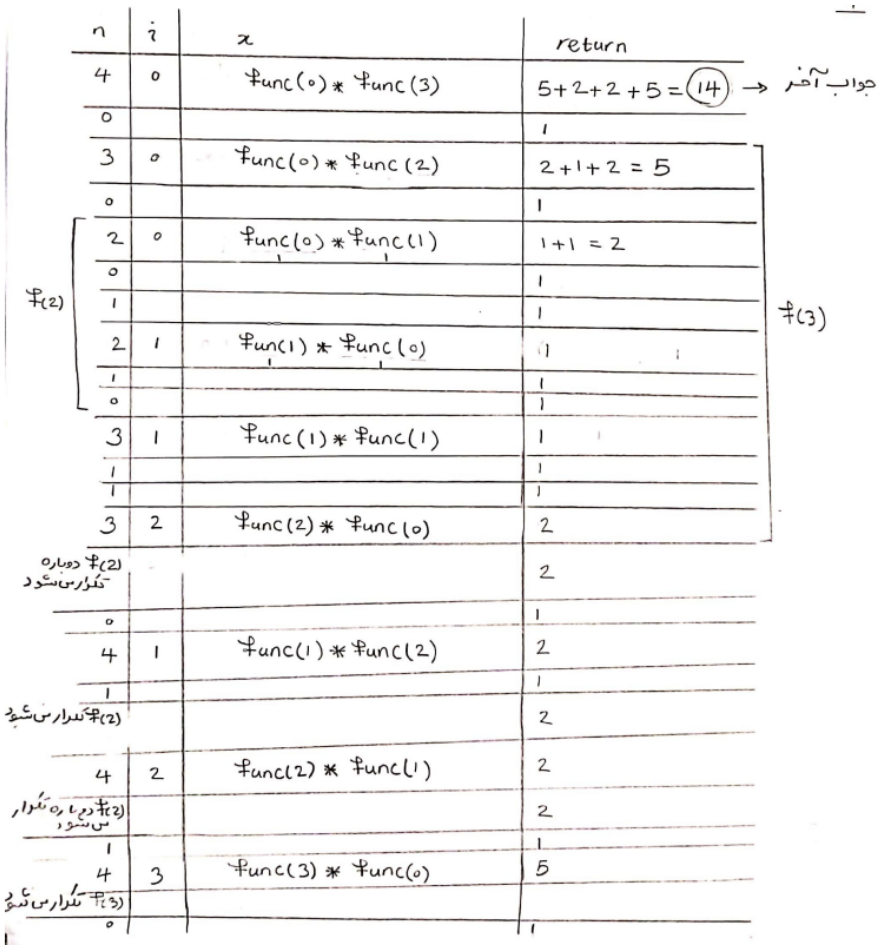
$$\log n\sum_{i=\frac{n}{2}}^{n}n-i+1 \implies (\frac{n^2}{8}+\frac{n}{4}+\frac{n}{2}+1)\log n \implies \mathcal{O}(n^2\log n)$$

سوال ۴

شبه کد زیر را به ازای n=۴ پیمایش (trace) کنید.

```
def func(n):
        if n == 0 or n == 1:
                return 1
        x = 0
        for i in range(n):
                x += func(i) * func(n-1-i)
        return x
```

پاسخ:



سوال ۵

تعداد تکرار جمله write(۱*۱) را در شبه کد زیر بیابید.

```
i = n
while i>1 do
        j = 1
        while j<n do
                j = j*3
                i = i/2
                write(1*1)
```

پاسخ:
می خواهیم بدانیم در هر بار تکرار حلقه بیرونی i چند بار تقسیم بر ۲ می شود.

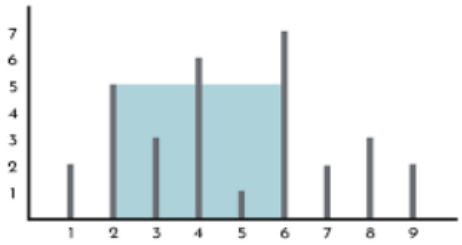
$$2 * 2 * ... * 2 \implies 2^{\log_3 n}$$

i در هر بار اجرای حلقه بیرونی بر $2^{\log_3 n}$ تقسیم می شود تا وقتی که $i \leq 1$ شود.
بنابر این حلقه $\log_{2^{\log_3 n}} n$ بار اجرا می شود. و تعداد اجرای write

$$\log_{2^{\log_3 n}} n * \log_3 n \implies \frac{1}{\log_3 n} * \log_2 n * \log_3 n \implies \log_2 n$$

سوال ۶

نموداری مانند شکل زیر در اختیار داریم و ارتفاع هر کدام از نمودار ها مشخص است. الگوریتمی ارایه کنید تا بتوان با استفاده از دو نمودار و محور x یک محفظه تشکیل بدهیم که بتواند بیشترین مقدار آب ممکن را ذخیره کند. شبه کد آن را بنویسید و آن را از نظر هزینه زمانی بررسی کنید.



مثال

ورودی : $[2, 5, 3, 6, 1, 7, 2, 3, 2]$
خروجی : $[2, 6]$

پاسخ:
نمودار را می توان با آرایه اینگونه پیاده سازی کرد که ارتفاع هر نمودار $A[i]$ باشد و شماره هر نمودار روی محور x برابر با i باشد و باید بیشترین مقدار را بدست آورید. برای حل این مساله باید از دو اشاره گر استفاده کرد. به عنوان مثال اشاره گر چپ و اشاره گر راست، اشاره گر چپ در ابتدای آرایه قرار می گیرد و اشاره گر راست در انتهای آرایه و باید عدد کوچکتر بین مقدار اشاره گر راست و اشاره گر چپ انتخاب شود و در (فاصله راست - چپ) ضرب شود. سپس با مقدار حداکثر مقایسه بشود. و اگر بزرگتر از حداکثر بود ذخیره شود. در ادامه اگر مقدار اشاره گر چپ کوچکتر از راست بود، اشاره گر به عقب حرکت می کند و در غیر این صورت اشاره گر چپ به جلو حرکت می کند. پیچیدگی زمانی الگوریتم از $\mathcal{O}(n)$ می باشد.

```
func (arr):
        left = 0 , right = n ,result=0
        while left < right :
                area = (right - left) * min(arr[right],arr[left])
                result = max(area,result)
                if arr[left]<arr[right]:
                        left++
                else:
                        right--
        return result
```

سوال ۷

در جنگلی بزرگ، غاری وجود داشت که توسط خرس های جنگل کشف شد. اما خرس ها نمی توانستند تصمیم بگیرند که چه کسانی باید در غار زندگی کند. بنابر این، به این نتیجه رسیدند که هر خرس به نوبت وارد غار شود. هر خرسی که وارد شود، خرس هایی را که قدرتشان کمتر یا برابر او هستند را از بین می برد و کنار خرس های قویتر از خود می ماند. او در غار منتظر خرس های بعدی می ماند تا با آن ها روبرو شود و قدرتشان را به چالش بکشد. الگوریتمی ارایه دهید که مشخص کند در انتها کدام یک از خرس ها در غار زنده می مانند. همچنین شبه کد آن را بنویسید و زمان اجرای آن را تحلیل کنید.

مثال

ورودی : $[1, 4, 5, 3, 1, 2]$
خروجی : $[5, 3, 2]$

پاسخ:
غار را می توان یک پشته در نظر گرفت و برای حل مساله از آن استفاده کرد. قدرت هر خرس را مقدار اندیس های آرایه در نظر می گیریم. در ابتدا که پشته خالی است اولین مقدار اندیس آرایه در پشته درج می شود[۱] و در ادامه مقدار اندیس دوم آرایه با عنصر بالای پشته [۲] مقایسه می شود، اگر عنصر بالای پشته کوچک تر و یا مساوی عدد مورد نظر باشد از پشته خارج می شود، در غیر این صورت عدد در پشته درج می شود. پیچیدگی زمانی الگوریتم از $\mathcal{O}(n)$ می شود.

```
func(arr)
        if stack is empty:
                stack.push(arr[0])
        for i=1 to n:
                while arr[i]> stack.topElement
                        stack.pop()
                stack.push(arr[i])
        return stack
```

[۱] push
[۲] top