سوال ۱

آرایه ای از اعداد صحیح مثبت به همراه یک عدد صحیح مثبت به عنوان هدف داده شده است. تمام ترکیبات ممکن را پیدا کنید که مجموع عناصر در ترکیب برابر با هدف باشد. آرایه داده شده دارای عناصر تکراری نیست و هر عنصر می تواند چندین بار انتخاب شود. الگوریتم خود را ارایه دهید و برای الگوریتم مورد نظر شبه کد بنویسید. همچنین هزینه زمانی آن را محاسبه کنید.

مثال

input : $[3, 4, 5]$
target : 9
output : $[[3, 3, 3],[4, 5],[5, 4]]$

پاسخ:

از ساختمان داده درخت استفاده می کنیم، زیرا می توان مسیر های متفاوتی را ایجاد کرد و هر مسیر می تواند نشان دهنده یک حالت خاص باشد.
روش حل به این صورت است که ریشه درخت برابر است با هدف و فرزندان ریشه می شود اعدادی که به عنوان ورودی به ما داده شده است و به همین ترتیب فرزندان هر گره برابر است با اعداد ورودی. از ریشه شروع می کنیم و مقدار فرزندانش از مقدار خودش کم می کنیم و سه حالت به وجود می آید

(۱) عددی بزرگ تر از صفر به دست می آید: مسیر را ادامه می دهیم.

(۲) عدد برابر با صفر است: یعنی مجموع گره ها برابر هدف است.

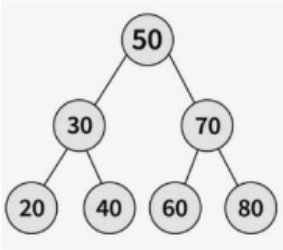
(۳) عدد کوچک تر از صفر به دست می آید: توقف می کنیم و دیگر آن مسیر را ادامه نمی دهیم.

در نهایت مسیر هایی که به صفر رسیدند را به عنوان جواب در نضر می گیریم و پیچیدگی زمانی آن برابر است با : $\mathcal{O}(n^k)$ k عمق درخت است

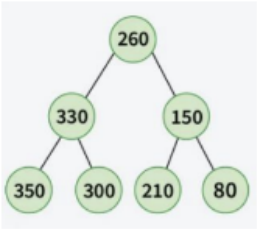
سوال ۲

یک درخت جستجوی دودویی داده شده است. الگوریتمی ارایه کنید که برای هر گره، مجموع مقادیر بزرگتر یا مساوی آن گره را به مقدار گره اضافه کند. شبه کد الگوریتم را بنویسید و آن را از نظر هزینه زمانی تحلیل کنید.

Input:



output:



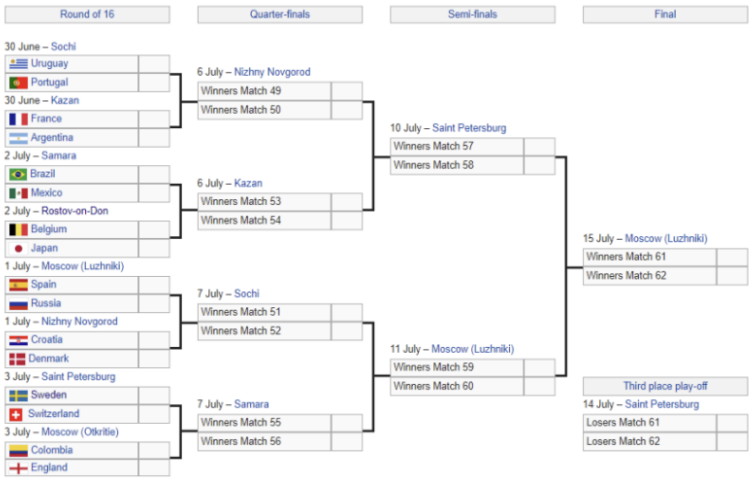
پاسخ:

با توجه به این که درخت داده شده یک درخت جستجوی دودویی است، بزرگترین عدد در آخرین فرزند راست زیر درخت راست قرار دارد. برای حل سوال می توان از بزرگترین گره شروع کرد و با گره های کوچک تر از خودش جمع کرد. پیمایش به این صورت خواهد بود:
زیر درخت راست - ریشه - زیر درخت چپ هزینه زمانی از $\mathcal{O}(n)$ خواهد بود.

```
greater_value(root):
        sum=0
        func(node):
                if node is None:
                        return
                func(node.right)
                temp=node.data
                node.data+=sum
                sum+=temp
                func(node.left)
        func(root)
        return root
```

سوال ۳

در یک تورنمنت فوتبال به سبک جام جهانی، تیم ها به صورت دو به دو با هم رقابت می کنند. در هر مرحله، تیم های برنده به مرحله بعد صعود می کنند و این روند تا مشخص شدن تیم قهرمان ادامه می یابد. فرض کنید برای هر بازی، تعداد کل گل های زده شده ثبت شده باشد. میخواهیم k امین مرحله ای را پیدا کنیم که در آن بیشترین گل ها به ثمر رسیده باشد. شبه کد الگوریتم خود را بنویسید و آن را از نظر هزینه زمانی تحلیل کنید.



پاسخ:

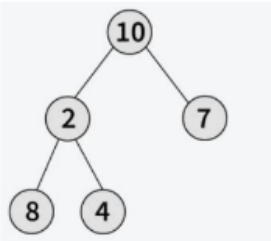
هر مرحله نشان دهنده ی یک سطح است. برای حل این سوال مجموع داده های هر سطح را محاسبه می کنیم. سپس k امین بیشتر مقدار را به دست می آوریم.
هزینه زمانی از $\mathcal{O}(n \log n)$ خواهد شد.

```
func (root, k)
        level_sums = Array()
        queue= deque(root)
        i=0
        while queue :
                current_level_sum=0
                for i in range(len(queue)):
                        node= queue.pop()
                        current_level_sum+=node.data
                        if node.left:
                                queue.push(node.left)
                        if node.right:
                                queue.push(node.right)
                level_sum[i]=current_level_sum
                i++
        if len(level_sum)<k:
                return
        level_sum.sort()
        return level_sum[n-k]
```

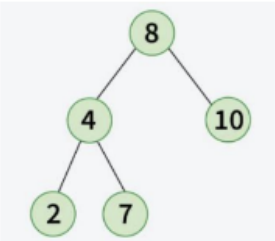
سوال ۴

یک درخت دودویی داده شده است و از شما خواسته شده آن را به یک درخت جستجوی دودویی تبدیل کنید. تبدیل باید به گونه ای انجام شود که ساختار اصلی درخت دودویی حفظ شود. شبه کد الگوریتم خود را بنویسید و آن را از نظر هزینه زمانی تحلیل کنید.

Input:



output:



پاسخ:

درخت داده شده را پیمایش می کنیم و مقدار هر گره را در یک آرایه ذخیره می کنیم. سپس آرایه را مرتب می کنیم و پیمایش میان ترتیب را اجرا می کنیم و مقدار هر گره را در داده های آرایه جایگذاری می کنیم. پیچیدگی زمان از $\mathcal{O}(n \log n)$ خواهد بود.

```
preorder(node,arr):
        if node is none:
                return
        arr.add(node.data)
        preorder(node.left)
        preorder(node.right)

conversion(node,arr,index):
        if node is none:
                return
        conversion(node.left,arr,index)
        node.data=arr[index]
        index+=1
        conversion(node.right,arr,index)

binary_to_BST(root):
        arr=Array()
        preorder(root,arr)
        arr=arr.sort()
        index=0
        conversion(root,arr,index)
        return root
```

سوال ۵.۱

پیمایش های preorder و postorder یک درخت دودویی کامل داده شده است. پیمایش inorder این درخت را بدست آورید. الگوریتم خود را بنویسید و برای الگوریتم مورد نظر شبه کد ارایه کنید. همچنین هزینه زمانی آن را محاسبه کنید.

: preorder

P،S،O،D،I،Z،J،H،B،Q،G،A،M،C،N

: postorder

O،Z،J،I،H،D،S،Q،A،C،N،M،G،B،P

پاسخ:

inorder: O,S,Z,I,J,D,H,P,Q,B,A,G,C,M,N

هزینه زمانی از $\mathcal{O}(n)$ خواهد بود.

```
func(pre,pos):
        if pre.size==0:
                return
        if pre.size==1:
                return root
        for i in range(pre.size):
                if pre[1]==post[i]:
                        root.left=func(pre[1:i+2],post[:i+1])
                        root.right=func(pre[i+2:],post[i+1:])
                        return root
```

سوال ۵.۲

الگوریتم ترتیبی برای پیمایش های preorder ، inorder و postorder ارایه کنید و شبه کد آن را بنویسید.

پاسخ:

```
iterative_inorder(node):
        i=0
        S=Stack()
        inorder=Array()
        while True:
                if node:
                        S.push(node)
                        node=node.left
                else:
                        if S is empty:
                                break
                        node=S.pop()
                        inorder[i]=node.data
                        i++
                        node=node.right
        return inorder
```

```
interative_postorder(node):
        postorder=Array()
        i=0
        if node is none:
                return postorder
        S1=Stack()
        S2=Stack()
        while S1 is not empty:
                node=S1.pop()
                S2.push(node)
                if root.left:
                        S1.push(node.left)
                if root.right:
                        S1.push(node.right)
                while S2 is not empty:
                        postorder[i]=S2.top
                        i++
                        S2.pop()

        return postordder
```

```
interative_preorder(node):
        preorder=Stack()
        if root is none:
                return preorder
        S=Stack()
        S.push(node)
        while S is not empty:
                node=S.pop()
```

```
۹                       preorder.push(node.data)
۱۰              if node.right:
۱۱                      S.push(node.right)
۱۲              if node.left:
۱۳                      S.push(node.left)
۱۴              return preorder
```

سوال ۶

اعداد ۱ تا ۱۷ را به ترتیب در یک درخت جستجوی دودویی درج کنید. پس از درج میخواهیم این درخت را به یک درخت جستجوی دودویی متوازن تبدیل کنیم. ارتفاع درخت در حالت متوازن چقدر است؟ به چند چرخش نیاز است تا درخت متوازن شود؟ تمامی چرخش ها را مرحله به مرحله نمایش دهید و الگوریتم خود را با جزییات شرح دهید.

پاسخ: