به نام هستی بخش

امتحان میان‌ترم

ساختمان‌داده‌ها و الگوریتم‌ها

مدت امتحان :۹۰ دقیقه – هشام فیلی – آذرماه ۱۴۰۱

نام و نام‌خانوادگی .....

شماره دانشجویی .....

نمره اخذ شده	بارم کل	بارم هر سوال	تعداد سوالات	نوع سوالات
	٤٥	۵ پاسخ منفی = منفی یک نمره	۹	چهارگزینه ای
	۳۵	۵	۷	جای خالی
	٤٠	۱۰	٤	پاسخ تشریحی
	۱۲۰			نمره نهایی

به نکات زیر توجه کنید:

۱. پاسخ هر سوال را در همین برگه و در محل تعیین شده بنویسید درصورت نیاز می‌توانید از پشت آخرین برگه نیز برای پاسخگویی استفاده کنید.

۲. لطفا تلفن همراه خود را همین الان خاموش کنید. هر گونه استفاده از تلفن همراه (یا تماس حین امتحان) به منزله تقلب است.

۳. امتحان ۱۲۰ امتیاز دارد که ۱۲۰ امتیاز آن بصورت کمکی در نظر گرفته‌شده است و نمره شما درنهایت از ۱۰۰ محاسبه خواهد شد.

سوالات چهارگزینه ای (بارم هر سوال سه نمره – هر پاسخ نادرست یک نمره منفی دارد)

1. در کدام گزینه، توابع بر اساس پیچیدگی زمانی به درستی مرتب شده‌اند؟

a. $log^2(n) \quad < n < \log(n!) < (\log(n))^{\log(n)}$

b. $n^2 < \sum_{i=0}^{\infty} \frac{n^i}{i!} < 2^n < n!$

c. $log^* n < \log(n) < (\log(n))! < n^2$

d. $log(n!) < (log(n))! < \sum_{i=1}^{n} i^3 < \sum_{i=1}^{n} i2^i$

پاسخ: گزینه ۱

2. پیچیدگی زمانی الگوریتم زیر کدام است؟

```
int i = n;
while (i > 1) {
        i /= 2;
        j = i;
        while (j > 1) {
                j /= 7;
        }
}
```

1) $O(nlog(n))$
2) $O((log(n))^2)$
3) $O(oglog(n))$
4) $O(log(n))$

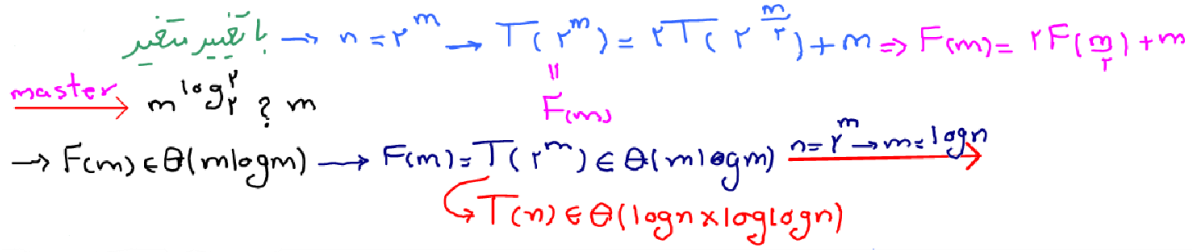
پاسخ: گزینه ۲ : اثبات: هربار که حلقه بیرونی iام اجرا می‌شود، حلقه داخلی به اندازه $log_7 i$ بار اجرا می‌شود. از آنجایی که مبنای لگاریتم در مجاسبه‌ی زمان اجرا مهم نمی‌باشد، مبنای تمام لگاریتم‌ها را ۲ در نظر می‌گیریم و کل زمان اجرا به صورت زیر محاسبه می‌شود:

$$log\frac{n}{2} + log\frac{n}{4} + \ldots = (logn)^2 - (1 + 2 + \ldots + logn) = (logn)^2 - \frac{(logn)(logn+1)}{2} = O((logn)^2)$$

3. پیچیدگی زمانی رابطه‌ی بازگشتی $T(n) = 2T([\sqrt{n}]) + logn$ کدام گزینه می‌باشد؟

1) $O(loglogn)$
2) $O(logn\ loglogn)$
3) $O(n\ loglogn)$
4) $O(n\ logn\ logn)$

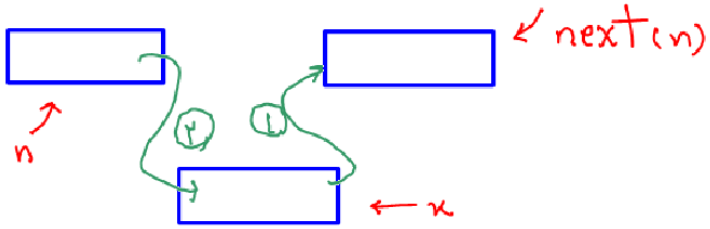
پاسخ: گزینه ۲



4. فرض کنید گره x باید بعد از گره n در یک لیست پیوندی یک‌طرفه درج شود. کدام گزینه به درستی اشاره‌گرها را مقداردهی می‌کند؟ (ترتیب عملیات‌ها از چپ به راست است و فرض کنید $next[n]$ وجود دارد)

1) $next[n] = x;\ next[x] = next[n]$
2) $next[n] = x;\ next[x] = next[next[n]]$
3) $next[x] = n;\ next[n] = x$
4) $next[x] = next[n];\ next[n] = x$

پاسخ: گزینه ۴



5. فرض کنید صف Q با یک آرایه‌ی حلقوی به اندازه m پیاده‌سازی شده است که اندیس‌های آن از ۰ تا m-۱ است و عناصر آن به صورت چرخه‌ای و در جهت ساعت‌گرد ذخیره شده‌اند. مولفه‌های $front(Q)$ و $rear(Q)$ به ترتیب اندیس اولین عنصر و عنصر بعد از آخرین عضو صف Q را ذخیره می‌کنند. تعداد عناصر داخل صف و شرط پر بودن صف به ترتیب کدام گزینه زیر است؟

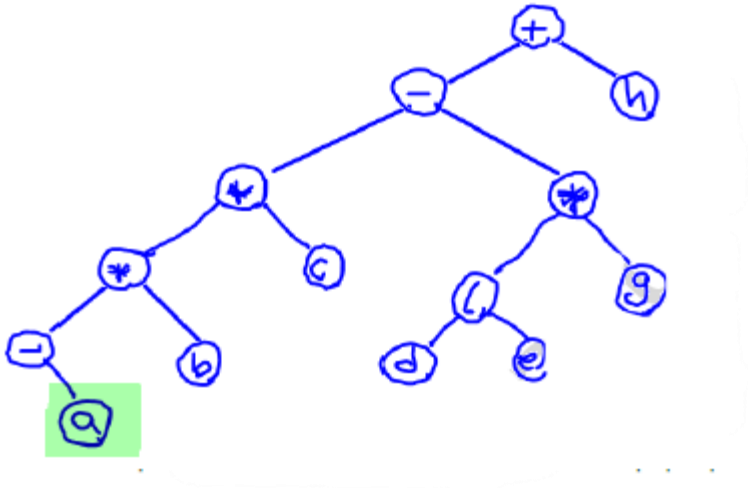
1) $front(Q) = rear(Q), rear(Q) - front(Q) + 1 mod m$
2) $front(Q) = rear(Q), rear(Q) - front(Q) mod m$
3) $front(Q) = rear(Q) + 1 mod m, rear(Q) - front(Q) + 1 mod m$
4) $front(Q) = rear(Q) + 1 mod m, rear(Q) - front(Q) mod m$

پاسخ: گزینه ۴

6. عمق درخت دودویی معادل با عبارت محاسباتی $(-a) * b * c - d/e * g + h$ برابر است با:

1) 4
2) 5
3) 6
4) 7

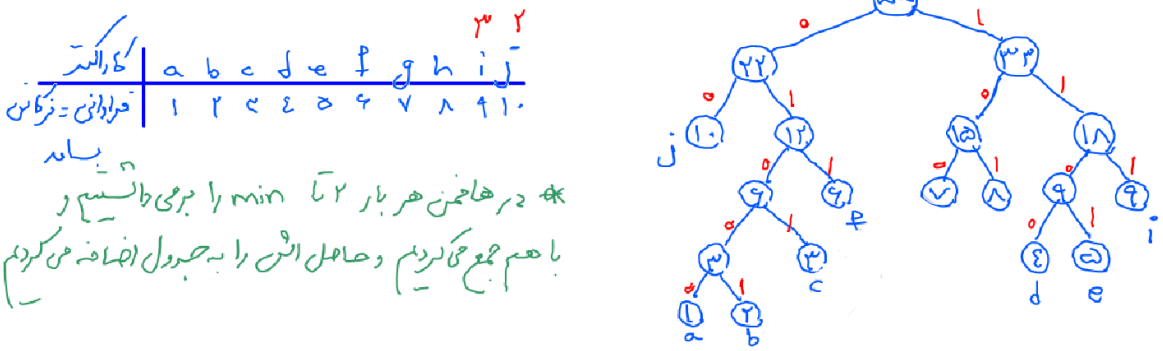
جواب: گزینه ۲ برای این که برای یک عبارت محاسباتی، درخت دودویی بکشیم، باید آن را به صورت infix دربیاوریم. با توجه به اولویت عملگرها داریم:



7. ارتفاع درخت هافمن اگر ورودی ۱۰ نشانه با بسامدهای ۱ تا ۱۰ باشد، چقدر است؟

1) 3
2) 4
3) 5
4) 6

جواب: گزینه ۳



8. کدام آرایه زیر می‌تواند نمایشگر یک درخت max-heap باشد؟

1) 14 8 10 13 16 12 25
2) 14 8 10 13 16 12 12
3) 12 8 10 13 16 14 25
4) 16 8 10 13 12 14 15

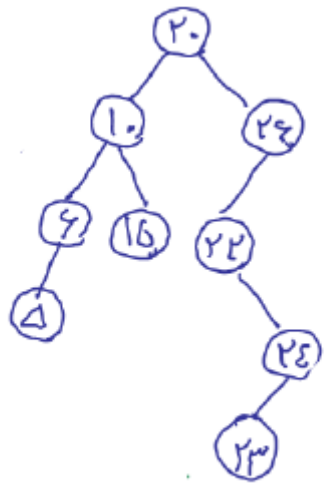
جواب: گزینه ۳ - در آرایه‌ای که از ایندکس ۰ شروع می‌شود، باید هر خانه را با خانه‌های $2i+1$ و $2i+2$ مقایسه کنیم. یعنی فرزند چپ خانه iام در $2i+1$ و فرزند راست آن در خانه $2i+2$ می‌باشد. با توجه به این که درخت max-heap است و مقدار موجود در هر گره از فرزندانش بزرگ‌تر است، فقط گزینه ۳ درست می‌باشد.

9. کدام گزینه پیمایش Preorder یک درخت جستجوی دودویی با پیمایش Postorder به صورت زیر است؟

Postorder: 5, 6, 15, 10, 23, 24, 22, 26, 20

1) 22 ,24 ,26 ,10 ,6 ,5 ,15 ,23 ,20
2) 26 ,23 ,24 ,22 ,20 ,15 ,10 ,5 ,6
3) 22 ,23, 24 ,26 ,15 ,5 ,6 ,10 ,20
4) 23 ,24 ,22 ,26 ,15 ,5 ,6 ,10 ,20

جواب: گزینه ۴



سوالات لطفا جای خالی را با گزاره یا پاسخ مناسب تکمیل کنید

1. مرتب شده ی توابع $\log n$، $(\log n)!$، n^2، $\sqrt{\log(n)}$، 2^n و $\log(\log(n))$ برحسب درجه به صورت میباشد.

$2^n > (\log n)! > n^2 > \log n > \sqrt{\log(n)} > \log(\log(n))$

2. پیچیدگی زمانی رابطه بازگشتی $T(n) = 125T\left(\frac{n}{5}\right) + 5n^3$ به صورت . . . . . میباشد.

با استفاده از قضیه اصلی، $f(n) = \theta(n^{\log_b a})$ است، پس در نتیجه جواب نهایی برابر $T(n) = \theta(n^{\log_b a} * \log n)$ یعنی $T(n) = \theta(n^3 * \log n)$ است.

3. زمان اجرای کد روبه رو به صورت . . . . . میباشد.

```
for( i = n; i > 1; i = i/3)
   for( j = i; j < n; j = j*2)
        for( k = 0; k < n; k += 2)
             //O(1)
```

مرتبه زمانی بیرونی ترین حلقه و حلقه میانی به صورت $O(\log(n))$ و درونی ترین حلقه به صورت $O(n)$ میباشد. پس مرتبه زمانی کلی برابر $O(n * \log^2 n)$ میباشد.

4. یک مزیت لیست پیوندی دوطرفه نسبت به لیست پیوندی یک طرفه، . . . . . و یک مزیت لیست پیوندی یک طرفه نسبت به لیست پیوندی دوطرفه، . . . . . است.

پیمایش راحت تر – استفاده از حافظه کمتر(موارد دیگری نیز میتواند باشد.)

5. حداقل و حداکثر تعداد گره‌های یک درخت دودویی کامل(درخت دودویی کامل به درختی گفته میشود که هر گره آن یا فرزندی ندارد یا دو فرزند دارد) با ارتفاع h برابر . . . . . و . . . . . میباشد.

(2h+1) و $(2^{(h+1)} - 1)$

6. نمایش پسوندی(postfix) عبارت $(A + B) * D + E/(F + A * D)$ به صورت . . . . . میباشد. (راهنمایی: ابتدا برحسب اولویت عملگرها، پرانتز گذاری کنید و سپس عبارت میانوندی داده شده را به عبارت پسوندی تبدیل کنید.)

$$AB + D * EFAD * +/+$$

7. یک درخت دودویی با 30 گره، اگر 10 گره دو فرزندی داشته باشد، تعداد گره های تک فرزندی برابر . . . . . است.

در هر درخت دودویی تنها گره های مرتبه دو(n_2)، یک(n_1) و صفر(n_0) داریم. با توجه به اینکه تعداد گره های مرتبه صفر(برگ ها)، یکی بیشتر از گره های مرتبه دو است($n0 = n2 + 1$)، پس خواهیم داشت:

$$n0 = 10 + 1 = 11$$
$$n1 = 30 - 10 - 11 = 9$$

سوالات با پاسخ های تشریحی (سعی شود که پاسخها تا حدالامکان مفید و مختصر باشند)

1. برای قطعه کد های بازگشتی زیر پیچیدگی زمانی را محاسبه کنید و پاسخ نهایی خود را توجیح کنید.

.a

```
void recursive(int a, int b, int c){
     if (a <= 0)
               return;
     recursive(a - 1, b + 1, c);
     recursive(a - 1, b, c + 1);
}
```

پاسخ: این الگوریتم صرفا روی a اجرا میشود و ربطی به b,c ندارد.
$T(a) = 2T(a-1)+1$ مشابه هانوی است $= 2^n-1$

.b

```
int func2(int m){
     if (m <= 0)
     return 1;
```

```
    else
    return 1 + 2*func2(m - 1);
}
```

پاسخ:

$T(m) = T(m-1) + 1 = O(m)$

2. الگوریتمی با پیچیدگی زمانی $O(n)$ طراحی کنید که در یک لیست پیوندی یک طرفه مرتب شده تمام جفت عضو هایی که مجموع آنها برابر عدد ثابت X می شود را بیابید.(استفاده از حافظه برای اشاره گر اضافه مجاز است. راهنمایی: از یک پشته استفاده کنید)

پاسخ:
از یک پشته برای ذخیره کردن اشاره گر به اعضای لیست پیوندی کمک می گیریم. p را اشاره گر به سر لیست پیوندی اختصاص می دهیم و q برابر مقدار سر پشته است.

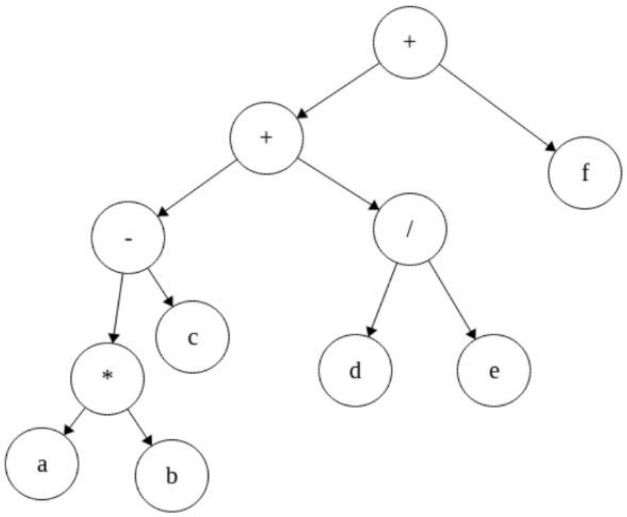
1- p را وارد پشته می کنیم و آن را یک خانه به جلو می بریم.
2- مقدار p و q را مقایسه می کنیم

الف) اگر $p + q = X$ باشد p و q یک جفت از جواب مسئله است پس در خروجی چاپ کرده و q را از پشته pop می کنیم و به مرحله ۱ می رویم.
ب) اگر $p + q > X$ باشد q را از پشته حذف کرده و به مرحله ۲ می رویم.
ج) اگر $p + q < X$ باشد به مرحله ۱ می رویم.
این کار را تا جایی ادامه می دهیم که به انتهای لیست یک طرفه برسیم. لیست یک طرفه تنها یک بار پیمایش شد پس از $O(n)$ است.

3. درخت عبارت برای عبارت زیر بکشید و نمایش postfix آن را هم بنویسید.

$a \times b - c + d/e + f$

پاسخ:
نمایش postfix: a b × c – d e / + f +



4. درخت جستجوی دودویی داریم که میخواهیم آن را به درخت max heap تبدیل کنیم. الگوریتمی با مرتبه زمانی $O(n)$ طراحی کنید که اینکار را انجام دهد. (n تعداد گره های درخت است.)

پاسخ:
درخت را به صورت inorder می پیماییم و مقدار هر گره را به ترتیب پیمایش در آرایه ای به طول n می ریزیم. به دلیل خصوصیت درخت BST که زیر درخت چپ کوچکتر از ریشه و زیر درخت راست بزرگتر از ریشه است پیمایش inorder باعث می شود آرایه بدست آمده سورت شده باشد. حال درخت را به صورت postorder می پیماییم و اعضای آرایه به ترتیب در گره ها می نویسیم.

```
def inorder(node):
    if not node:
        return
    inorder(node.left)
    copy_to_array(node.data)
    inorder(node.right)

def postorder(node):
    if not node:
        return
    postorder(node.left)
    postorder(node.right)
    node.data = copy_from_array()
```