1) گزاره های زیر را با استفاده از تعاریف اولیه ی O و ω اثبات کنید (2 نمره)

A) $3\sqrt{n}+5 \in O(n)$

B) $2^n - 100 \in \omega(n^3)$

---

2) پیچیدگی محاسباتی توابع زیر را محاسبه کنید. برای A از روش درخت بازگشت استفاده کنید. برای B از روش جایگذاری و تکرار استفاده کنید و برای C از قضیه ی مستر (Master Theorem) استفاده کنید. برای سادگی می توانید فرض کنید که n مثلاً توانی از دو یا توانی از سه می باشد. (2 نمره)

A) $T(n) = T(n/3)+T(2n/3) + 2n$, $T(1) = 1$, $T(2) = 1$.

B) $F(n) = F(n/2)+ \log_2(n)$, $F(1) = 1$.

C) $G(n) = 9G(n/3)+ n^2 \log_2(n)$, $G(1) = 1$.

---

3) داده ساختاری داریم که شامل k پشته (stack) می باشد و عملیات زیر را پشتیبانی می کند:

- push(x,i): این عمل در زمان $\Theta(1)$ عدد x را وارد پشته ی i ام می کند
- pop(i,j): این عمل در زمان $\Theta(i-j+1)$ عدد سر لیست های i ام تا j ام را خارج می کند
- halfpop(i): فرض کنید size(i) تعداد اعداد پشته ی i ام را نشان می دهد. این عمل در زمان $\Theta(size(i))$ نیمی از عددهای لیست i ام (یعنی [size(i)/2] تا) را خارج می کند.

الف) اگر n عمل از اعمال فوق را انجام دهیم پیچیدگی محاسباتی سرشکن (amortized) هر عمل را محاسبه کنید. توجه کنید که هیچ متغیری جز n در پیچیدگی محاسباتی ظاهر نشود (برای مورد ب و ج هم همینطور).

ب) فرض کنید این داده ساختار عمل زیر را هم پشتیبانی می کند. حال پیچیدگی محاسباتی سرشکن هر عمل چقدر است؟

- push(x,i,j): این عمل در زمان $\Theta(i-j+1)$ عدد x را در همه ی پشته های i ام تا j ام وارد می کند

ج) فرض کنید این داده ساختار عمل زیر را هم پشتیبانی می کند (ولی عمل push(x,i,j) را پشتیبانی نمی کند). حال پیچیدگی محاسباتی سرشکن هر عمل چقدر است؟

- push<(x,i): این عمل ابتدا تا زمانی که عدد سر پشته i ام از x کمتر است (و پشته ی i ام خالی نشده است) عدد سر پشته ی i ام را خارج می کند؛ سپس عدد x را وارد پشته ی i ام می کند. زمان مصرفی این عمل $\Theta(r+1)$ است که r تعداد اعداد خارج شده از پشته i ام در این مرحله است.

(مجموعا 2 نمره)

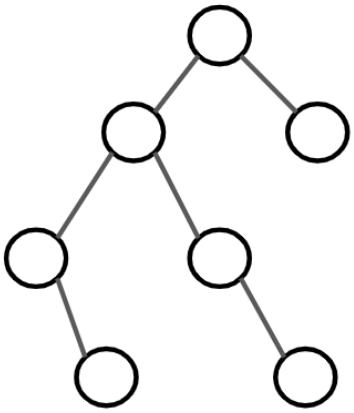
---

4) دنباله ای از n عدد به ترتیب رندوم (uniformly at random) داریم. در اینجا uniformly at random به این معنی است که احتمال این که هر ترتیبی دیده شود یکسان است. برای سادگی فرض کنید n بر 3 بخش پذیر است. حال این الگوریتم را در نظر بگیرید: به n/3 اعداد اول این دنباله ی اعداد نگاه می کنیم و بزرگترین این اعداد را پیدا می کنیم. نام این عدد را x می گذاریم. سپس به باقی دنباله نگاه می کنیم و

اولین عددی که **مساوی** x بود را انتخاب می کنیم. اگر در ادامه ی دنباله هیچ عددی مساوی x نبود هیچ عددی را انتخاب نمی کنیم.
الف) با فرض این که بزرگترین عدد دنباله دقیقا دو بار در این دنباله ظاهر شده است احتمال این که این الگوریتم بزرگترین عدد را انتخاب کند را تخمین بزنید. (1 نمره)
ب) با فرض این که بزرگترین عدد دنباله دقیقا k بار در این دنباله ظاهر شده است احتمال این که این الگوریتم بزرگترین عدد را انتخاب کند را بر حسب k تخمین بزنید. (1 نمره)

---

5) فرض کنید یک min-heap با n عنصر داریم و آن را در یک آرایه با index های 1 تا n ذخیره کرده ایم.
الف) index سمت راستی ترین برگ در این درخت چیست؟ توضیح دهید که چطور این index را حساب کردید.(توجه کنید که معمولا هم در لایه ی آخر هم در لایه ی یکی مانده به آخر درخت برگ وجود دارد). (1 نمره)
ب) عملگری با پیچیدگی زمانی بهینه طراحی کنید که عدد موجود در سمت راستی ترین برگ را به زیر درخت سمت چپ ریشه منتقل کند. توجه داشته باشید که بعد از اجرای این عملگر درخت همچنان باید min-heap بماند. پیچیدگی زمانی الگوریتم شما چقدر است؟ (2 نمره)



6) الف) اعداد مقابل را طوری در درخت مقابل قرار دهید که یک درخت جست و جو دودویی را تشکیل دهند (0.5 نمره): 8, 1, 2, 5, 12, 7, 3
ب) با استفاده از روش معرفی شده در کتاب عدد 6 را به این درخت جست و جو دودویی اضافه کنید. اعمال خود را مرحله به مرحله توضیح دهید. (0.75 نمره)
ج) سپس با استفاده از روش معرفی شده در کتاب عدد 3 را از این درخت حذف کنید. اعمال خود را مرحله به مرحله توضیح دهید. (0.75 نمره)