هر سوال را در محل در نظر گرفته شده پاسخ دهید. پاسخ های خارج از محل تصحیح نمیشوند. میتوانید از پشت صفحه به عنوان پیش نویس استفاده کنید. نام و شماره دانشجویی را روی **تمام برگه ها** بنویسید. شماره دانشجویی باید با **اعداد لاتین** نوشته شود.

۱. [۴۲] پیچیدگی محاسباتی:

(آ) [۱۲] درستی یا نادرستی عبارات زیر را با علامت (✓) یا (✗) مشخص کنید. دلیل خود را در نقطه چین زیر هر عبارت توضیح دهید. نمره کامل فقط به جواب صحیح با توضیح صحیح تعلق میگیرد.

دیگه واقعا انتظار میره که دانشجوی رشته کامپیوتر معنای پیچیدگی محاسباتی Order و Big-O و مقایسه آنها را با هم مسلط باشد. من یک مقداری تعجب کردم که بعضی از دانشجویان سر امتحان هیچ ایده ای از Big-O نداشتند. حتما یه مطالعه ای بفرمایید. به طور کلی که معنی Order یا Asymptotic-Notation ها مقایسه توابع هنگامی که n به سمت بی نهایت میل کند است. در واقع مقایسه نرخ رشد توابع است. معنای $f(n) = \mathcal{O}(g(n))$ این است که برای n "به اندازه کافی بزرگ" $f(n)$ حداکثر $k \times g(n)$ است (k یک عدد ثابت). توضیحات بیشتر را در اسلاید ها یا اینجا ببینید.

i. $n = \mathcal{O}(n \log_2 n)$ ✓
برای n به اندازه کافی بزرگ n همیشه از $n \log_2 n$ کوچکتر است.

ii. $n \log_2 n = \mathcal{O}(n)$ ✗

iii. $n = \Theta(2^{\log_2 n})$ ✓
$n = a^{\log_a n}$

iv. $\log_2 n = \mathcal{O}(n \log_2 n)$ ✓

v. $n^2 = \mathcal{O}(n^3)$ ✓

vi. $n^2 = \Omega(2^n)$ ✗
تعریف Ω برعکس $\mathcal{O}$ است. n^a همیشه از b^n کوچکتر است به شرطی که n به اندازه کافی بزرگ باشد. برای هر a و b

vii. $1.01^n = \mathcal{O}(n^2)$ ✗
n^a همیشه از b^n برای هر a و b کوچکتر است به شرطی که n به اندازه کافی بزرگ باشد.

viii. $10^{n+50} = \Theta(10^n)$ ✓
$10^{n+50} = 10^{50} \times 10^n = k \times 10^n = \Theta(10^n)$

ix. $0.99^n = \mathcal{O}(\log_2 n)$ ✓
برای n به اندازه کافی بزرگ : $0.99^n \approx 0$

x. $n^2 = \mathcal{O}(10^{15} n)$ ✗
$10^{15} \times n = k \times n < n^2$

xi. $10^{n+50} = \Omega(10^n)$ ✓
$10^{n+50} = 10^{50} \times 10^n = k \times 10^n = \Omega(10^n)$

xii. $n^2 = \mathcal{O}(10^{15} n)$ ✗
تکراری

(ب) [۶] رابطه بازگشتی و پیچیدگی محاسباتی الگوریتم های زیر را بنویسید. n طول آرایه یا تعداد گره های یک درخت است.

i. Selection Sort: $T(n) =$ $T(n-1) + \mathcal{O}(1)$, $\mathcal{O}($ n^2 $)$

ii. Merge Sort: $T(n) =$ $2 \times T(n/2) + \mathcal{O}(n)$, $\mathcal{O}($ $n \log_2 n$ $)$

iii. Calculating height of a binary tree: $T(n) =$ $2 \times T(n/2) + \mathcal{O}(1)$, $\mathcal{O}($ n $)$

(ج) [۸] پیچیدگی محاسباتی $\mathcal{O}($____n____$)$ و حافظه ای $\mathcal{O}($____$\log_3 n$____$)$ متد $F(n)$ چیست؟ پیچیدگی محاسباتی $\mathcal{O}($____n^2____$)$ و حافظه ای $\mathcal{O}($____n____$)$ متد $G(n)$ چیست؟ دلیل خود برای هر یک از این چهار مورد را بطور مختصر در نقطه چین زیر توضیح دهید. دقت کنید که پیچیدگی حافظه شامل Stack و Heap میشود.

```
1   public static int G(int n)
2   {
3       int sum = 0;
4       // Assumption: LinkedList is a singly linked  list data structure.
5       // It maintains a pointer to the last element.
6       LinkedList<int> s = new LinkedList<int>();
7       for(int i=0; i<n; i++)
8           s.AddLast(F(i,n));
9
10      while(s.Count > 0)
11      {
12          sum += s.Last();
13          s.RemoveLast();
14      }
15
16      return sum;
17  }
18
19  public static int F(int min, int max)
20  {
21      int len = max - min + 1;
22
23      if (len <= 1)
24          return len;
25
26      int mid1 = min + (len / 3);
27      int mid2 = mid1 + (len / 3);
28
29      return F(min, mid1) + F(mid1+1, mid2) + F(mid2+1, max);
30  }
```

$n = max - min, F(n) = 3 \times F(n/3) + O(1) \implies F(n) = O(n)$ (Master Theorem)
Depth of recursion for F is $\log_3 n \implies F_{\text{Memory}}(n) = O(\log_3 n)$

پیچیدگی محاسباتی خط هشتم $O(n-i)$ است (اضافه کردن به لینکد لیست $O(1)$ است) که در یک حلقه n تایی صدا زده میشود. پس پیچیدگی محاسباتی خطوط ۷ و ۸ میشود:
$n \times (1 + 2 + ... + n) < n \times (\frac{n}{2} + (\frac{n}{2} + 1) + ... + n) = \frac{n^2}{2} + n \times k = O(n^2)$
خطوط دهم تا سیزدهم n بار اجرا میشوند که چون پیچیدگی محاسباتی حذف کردن آخرین عنصر از لیست یک طرفه $O(n)$ است پس پیچیدگی محاسباتی این خطوط هم میشود $O(n^2)$ و پیچیدگی محاسباتی متد G در مجموع میشود $O(n^2)$. نهایتا متد G علاوه بر صدا زدن متد F یک لینکد لیست دارد که به اندازه n فضا اشغال میکند، پس میزان حافظه مورد استفاده متد G میشود $\log_3 n + n = O(n)$.

(د) [۱۶] پیچیدگی محاسباتی عملیات های زیر را در بدترین حالت و حالت متوسط/سرشکن همراه دلیل آنها بنویسید.

i. Insert in a singly linked list: $\mathcal{O}_{\text{worst}}($____$1$____$)$ $\mathcal{O}_{\text{amortized}}($____$1$____$)$

اضافه کردن به لینکد لیست همیشه $O(1)$ است. در بهترین حالت، بدترین حالت و حالت متوسط. عنصر جدید را به ابتدا یا انتهای لیست اضافه میکنیم. برای توضیح بیشتر به اسلایدها مراجعه کنید.

ii. Delete in a singly linked list: $\mathcal{O}_{\text{worst}}($____$n$____$)$ $\mathcal{O}_{\text{amortized}}($____$n$____$)$

باید با $O(n)$ عنصر مورد نظر را پیدا کنیم و با $O(1)$ حذف کنیم. چنانچه اشاره گر نود را داشته باشیم باز هم باید به $O(n)$ نود قبلی را پیدا کنیم تا بتوانیم حذفش کنیم. بدترین حالت و حالت متوسط هم فرقی نمیکند.

iii. Insert in a sorted array: $\mathcal{O}_{\text{worst}}(\underline{\quad n \quad})$ $\mathcal{O}_{\text{amortized}}(\underline{\quad n \quad})$

مکان اضافه کردن را با $O(\log n)$ پیدا میکنیم. برای اضافه کردن باید همه عناصر را با $O(n)$ شیفت بدهیم. حالت متوسط و بدترین حالت هم فرقی نمیکند.

iv. Extract Max in a Max-Heap: $\mathcal{O}_{\text{worst}}(\underline{\quad \log n \quad})$ $\mathcal{O}_{\text{amortized}}(\underline{\quad \log n \quad})$

هیپ همیشه بالانس است. پس ارتفاع آن همیشه از $O(\log n)$ است. پیدا کردن مقدار بیشینه $O(1)$ است چون همیشه در ریشه است. جایگزین کردن و SiftDown هم برای حفظ هیپ با $O(\log n)$ انجام میشود. مجددا حالت متوسط و بدترین حالت یکی است.

v. Insert in a Min-Heap: $\mathcal{O}_{\text{worst}}(\underline{\quad \log n \quad})$ $\mathcal{O}_{\text{amortized}}(\underline{\quad \log n \quad})$

اضافه کردن با $O(1)$ انجام میشود و SiftUp هم با $O(\log n)$ برای حفظ هیپ انجام میشود. باز بدترین حالت و حالت متوسط فرقی نمیکند.

vi. Find in a Binary Search Tree: $\mathcal{O}_{\text{worst}}(\underline{\quad n \quad})$ $\mathcal{O}_{\text{amortized}}(\underline{\quad n \quad})$

بدترین حالت برای یک درخت دو دویی این است که به شکل یک لینکد لیست در آمده باشد که میشود $O(n)$. حالت متوسط هم بدون اضافه کردن محدودیت یا فرض های بیشتر نمیتوان اضافه کرد و $O(n)$ مورد قبول است. اگر فرض دیگری کرده باشید مثل بالانس بودن ... پیچیدگی محاسباتی متناسب با فرض شما ممکن است متفات باشد.

vii. Find in an AVL Tree: $\mathcal{O}_{\text{worst}}(\underline{\quad \log n \quad})$ $\mathcal{O}_{\text{amortized}}(\underline{\quad \log n \quad})$

درخت دو دویی AVL همیشه بالانس است و در بدترین حالت و حالت متوسط ارتفاعش $O(\log n)$ است. پس پیچیدگی محاسباتی پیدا کردن هم میشود $O(\log n)$.

viii. Find in an Splay Tree: $\mathcal{O}_{\text{worst}}(\underline{\quad n \quad})$ $\mathcal{O}_{\text{amortized}}(\underline{\quad \log n \quad})$

بدترین حالت SplayTree مانند بدترین حالت درخت دو دویی معمولی است ولی به خاطر انجام Splay در حالت متوسط یا سرشکن پیدا کردن با $O(\log n)$ انجام میشود.

۲. [۶] خروجی پیمایش های مختلف یک درخت.

(a) In Order

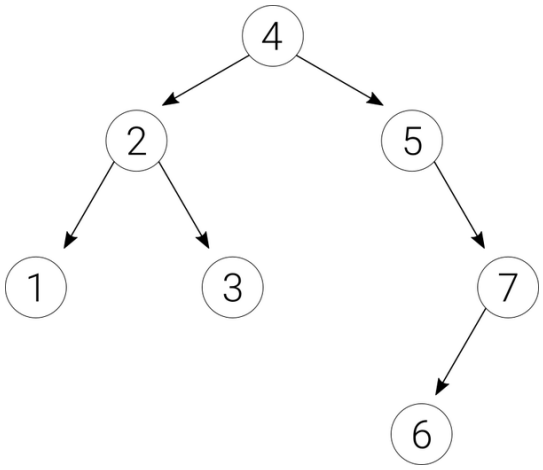
1 2 3 4 5 6 7

(b) Pre Order

4 2 1 3 5 7 6

(c) Post Order

1 3 2 6 7 5 4



۳. [۷] لیست زیر پیمایش PostOrder یک درخت باینری است. درخت مربوطه را رسم کنید. به منظور منحصر به فرد بودن جواب برای تمام گره هایی که بچه سمت چپ یا سمت راست ندارند، در محل پیمایش، برای هر فرزند نداشته، عدد منفی یک گذاشته شده.

-1 -1 -1 5 -1 -1 6 3 -1 -1 -1 7 4 2 1

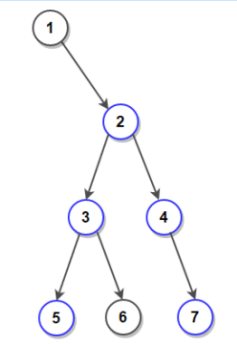
کد درست کردن این درخت را در کد مربوط به تمرین SplayTree داخل کلاس BST متد ParseBST برای PreOrder گذاشته بودم و منظورم از این سوال این بود که این الگوریتم بازگشتی را خودتان بتوانید طراحی و اجرا کنید. این کد Pre-Order است. برای Post-Order هم مشابه همین است.

```
public static Node ParseBST(ref IEnumerable<long> preOrderList)
{
    if (!preOrderList.Any())
        return null;

    long nextNode = preOrderList.First();
    preOrderList = preOrderList.Skip(1);

    if (nextNode == -1)
        return null;

    Node n = new Node(nextNode);

    n.Left = ParseBST(ref preOrderList);
    n.Right = ParseBST(ref preOrderList);

    return n;
}
```

نمونه استفاده در تست ها:

```
[TestMethod()]
public void InsertTest()
{
    IEnumerable<long> preOrderList = new List<long>() { 30, 20, 10, -1, -1, -1, -1 };
    var root = BST.ParseBST(ref preOrderList);
    ...
}
```

جواب:



۴. [۱۰] ورودی متد زیر یال های یک گراف بدون جهت است. خروجی آن چیست؟ پیچیدگی محاسباتی این متد بر حسب تعداد گره ها N و تعداد یال ها E چیست؟ $\mathcal{O}(\quad E \times \log^* N \quad)$ در نقطه چین زیر توضیح دهید.

```
// "edges" contains a list of edges for an undirected
// graph. Each edge is represnted by an array of size 2.
// The first element is the source node and the second
// element is the target node.
public static bool Solve(long[][] edges)
{
    DisjointSet ds = new DisjointSet(edges.Length);
    foreach(var edge in edges)
    {
        var x = ds.Find(edge[0]);
        var y = ds.Find(edge[1]);
```

```
            if (x == y)
            return true;
                ds.Union(x, y);
        }
        return false;
    }
```

پیچیدگی محاسباتی با فرض استفاده از تکنیک های Path-Compression و Union-by-Rank در DisjointSet میشود پیمایش تمام یال ها $O(E)$ ضرب در هزینه Find و Union در DisjointSet که میشود $\log^*(N)$ که برای n های مورد استفاده در اکثر کاربردها میتوان $\log^*$ را مقدار ثابت فرض کرد. اگر فرض دیگری در مورد DisjointSet کردید پیچیدگی محاسباتی Find و Union ممکن است متفاوت باشد.
این متد فقط درصورتی true برمیگرداند که در گراف دور داشته باشیم.

۵. [۱۰] پیچیدگی محاسباتی $\mathcal{O}(\frac{M+N}{})$ و حافظه ای $\mathcal{O}_{Memory}(\frac{N}{})$ الگوریتم Rabin-Karp بر حسب طول متن N و طول عبارت جستجو M چیست؟ کاربرد الگوریتم و نحوه کار آن را بصورت واضح و مختصر توضیح دهید. حتما به ایده اصلی این الگوریتم اشاره کنید.

کاربرد: برای پیدا کردن یک عبارت در یک متن.
نحوه کار: بجای مقایسه عبارت با تمام مکان ها در متن، hash عبارت با hash تمام مکان ها مقایسه میشود و تنها در صورت یکی بودن hash ها خود عبارت با رشته حرفی در متن مقایسه میشود. محاسبه hash برای یک عبارت به اندازه M هزینه $O(M)$ دارد و محاسبه آن برای تمام مکان ها در متن میشود $O(M \times N)$.
نکته اصلی این الگوریتم این است که تمام هاhash را با $O(M+N)$ حساب میکند. به این صورت که با استفاده از فرمول تابع hash برای هر مکان جدید از hash قبلی یک عبارت ریاضی کم میکند و یک عبارت ریاضی اضافه میکند که مربوط میشود به یک کاراکتر که اضافه شده و یک کاراکتر که کم شده. برای توضیح بهتر اسلایدها را ببینید. از نظر حافظه هم گرچه امکان بهینه سازی بیشتر موجود است ولی الگوریتم به صورتی که در کلاس و اسلاید ها عنوان شد برای نگهداری hash های از پیش محاسبه شده به $N-M$ حافظه و برای hash عبارت جستجو به M حافظه نیاز دارد که در مجموع میشود $O(N)$.

۶. [۱۰] ساختار داده ای Bloom-Filter چیست و چه کاربردی دارد(مثال بزنید)؟ مزیت اصلی این ساختمان داده ای بر ساختمان های داده ای مشابه چیست؟ این ساختار داده ای چه متغیرهایی دارد؟ در مورد پیچیدگی محاسباتی و حافظه ای آن توضیح دهید.

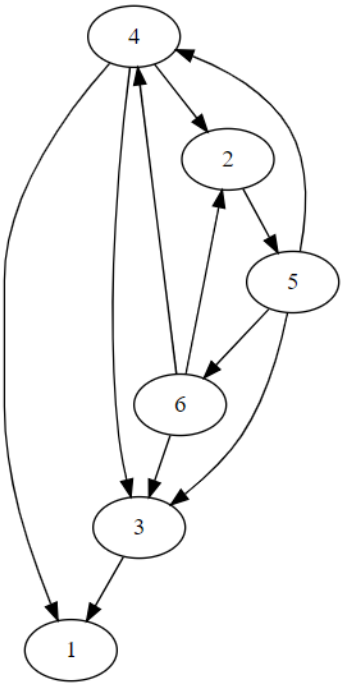
این ساختار داده ای دو عملگر Add(key) و Find(key) دارد. بدون اینکه عملگر Add کلیدها را ذخیره کند، عملگر Find میتواند بگوید که آیا کلیدی قبلا دیده (Add) شده یا نه. اگر بگوید دیده نشده، حتما دیده نشده، ولی اگر بگوید دیده شده با یک احتمالی ممکن است که دیده نشده باشد. این احتمال بستگی به پارامترها/متغیرهای این ساختمان داده ای دارد. پارامترها عبارتند از تعداد بیت های و تعداد توابع .hash این ساختمان داده از جهتی شبیه HashTable است. با این تفاوت که در HashTable خود کلیدها هم ذخیره میشوند و در هنگام collision کلیدها با هم مقایسه میشوند. برای بلوم فیلتر بجای ذخیره کلیدها از چند تابع hash استفاده میشود که احتمال collision در تمام توابع خیلی کم است. مزیت اصلی این ساختمان داده ای این است که نیازی به ذخیره کلیدها نیست و فضای بسیار کمتری نسبت به حجم کلیدها اشغال میکند. میتوانید یک مثال خوب و واقعی از یاهو میل را اینجا ببینید.

۷. [۱۵] در گراف زیر Strongly-Connected-Components را با نشان دادن مراحل پیدا کنید. توضیح لازم نیست. نشان دادن خروجی الگوریتم در مراحل مختلف لازم است.

DFS on reverse graph (starting at 1) $\Longrightarrow$ largest post order on reverse graph: 1.
DFS from 1 on original graph $\Longrightarrow$ first SCC: (1).

Remove (1) from graph. next largest post order (3). Do the same... second SCC: (3).

Remove (3). Next largest post order: (6). DFS from (6) ... third SCC:(6,4,2,5).



۸. [۱] لیست تمرین هایی را که در آنها بیش از پنج خط متوالی کپ زدید را بنویسید.

◯ A2 ◯ A3 ◯ A4 ◯ A5 ◯ A6 ◯ A7
◯ A8 ◯ A9 ◯ A10 ◯ A11 ◯ A12

۹. اگر انتقاد، پیشنهاد، کامنت، پیام، ... دارید، اگر بفرمایید ممنون میشم. در صورت کمبود جا از پشت همین صفحه استفاده کنید. با تشکر فراوان.