دانشکده مهندسی کامپیوتر

ساختمان‌های داده

امتحان عملی دوم

تهیه و تنظیم سوالات:
مبین داریوش همدانی
بابک بهکام کیا

استاد درس: سید صالح اعتمادی

نیم‌سال اول ۱۴۰۱-۱۴۰۰

نام شاخه	fb_E2
نام پروژه/پوشه/پول ریکوست	E2
مهلت ارسال	۱۳ دی ساعت ۱۴:۳۰

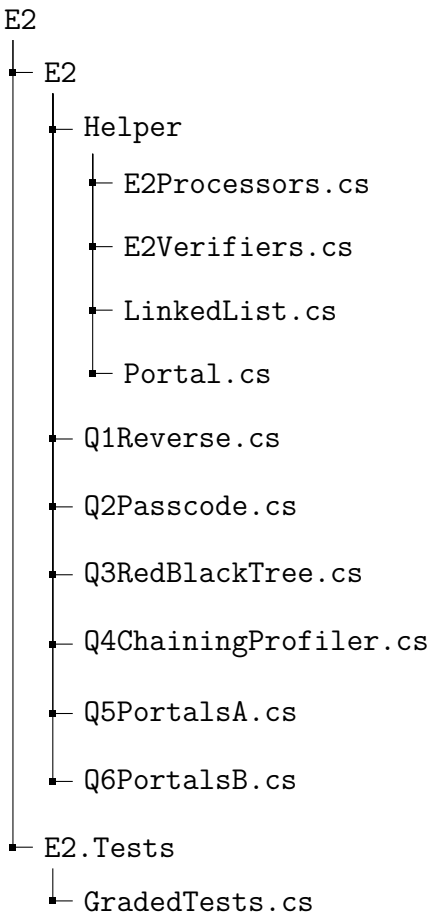
راهنمای امتحان

ساخت پروژه

۱. با اجرای اسکریپت ساخت پروژه در ریشه ریپازیتوری خود برای درس ساختمان داده، یک پروژه برای امتحان عملی با نام E2 بسازید: [۱]

```
dsproj -Create `
-cname E2 `
-testcommon .\TestCommon\TestCommon.csproj `
-testdata .\DS_E2\_publish\TestData\
```

۲. فایل‌های سوالات، تست، و همینطور فایل‌های کمکی را با توجه به ساختار زیر به پروژه خود اضافه کنید.



[۱] می‌توانید به صورت دستی نیز مراحل ساخت پروژه را انجام دهید. در این صورت حتما به نام پروژه و پوشه‌ها توجه داشته باشید که درست باشند.

سایر نکات

۱. استفاده از اینترنت فقط و فقط برای استفاده از نرم‌افزار Teams و درست کردن PullRequest و جستجو در مورد سینتکس یا Error Code مجاز است. چنانچه اشکالی در مورد امتحان داشتید با استاد/حل‌تمرین در محیط تیمز در میان بگذارید.

۲. استفاده از اسلایدهای درس و کدهایی که «خود شما» برای «این درس» نوشته و در گیت موجود دارید مجاز است. استفاده از هرگونه کد دیگر که یا توسط شما نوشته نشده یا در برای این درس نوشته نشده یا در گیت شما قبلا موجود نبوده مجاز نمی‌باشد.

۳. استفاده از هر گونه ویدیو مجاز نمی‌باشد.

۴. تصویر صفحه نمایش و وب‌کم شما در کل مدت امتحان بدون وقفه باید توسط نرم‌افزار FlashBackExpress (یا نرم‌افزار مشابه) ضبط شده و پس از فشرده‌سازی برای استاد درس ارسال شود.

۱ Reverse (۲۵ نمره)

در این سوال وظیفه شما معکوس کردن لیست پیوندی[۲] می‌باشد. در ورودی یک لیست پیوندی و طول آن به شما داده می شود. برنامه شما باید به عنوان خروجی یک لیست پیوندی برگرداند که ترتیب اعضای آن برعکس لیست ورودی باشد.

محدودیت ها

- $1 \le n \le 15000$
- $1 \le a[i] \le 40000$

نمونه ۱

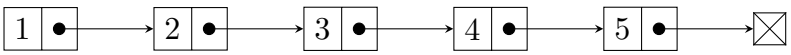
ورودی:

```
5
1
2
3
4
5
```

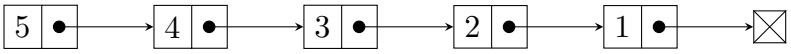
خروجی:

```
5 4 3 2 1
```

توضیح:
لیست پیوندی اولیه:



لیست پیوندی نهایی:



[2] Linked List

کلاس مسئله

```
using System;
using TestCommon;

namespace E2
{
    public class Q1Reverse : Processor
    {
        public Q1Reverse(string testDataName) : base(testDataName)
        {
        }

        public override string Process(string inStr)
            => E2Processors.ProcessQ1Reverse(inStr, Solve);

        public LinkedList<long> Solve(long n, LinkedList list)
        {
            throw new NotImplementedException();
        }
    }
}
```

۲ Passcode (۲۵ نمره)

به منظور بازیابی اطلاعات یک کامپیوتر قصد داریم که نام کاربری و رمز عبور آن را پیدا کنیم. می‌دانیم که نام کاربری و کلمه عبور این کامپیوتر هر کدام یک عدد طبیعی هستند، به طوری که حاصل ضرب کلمه عبور و نام کاربری برابر با x است. همچنین آرایه‌ای n عضوی به شکل $a_1, a_2, \cdots, a_n$ به ما داده شده است که حدس می‌زنیم کلمه عبور و نام کاربری از بین اعضای آن انتخاب شده اند.

برنامه‌ای بنویسید که بررسی کند که آیا چنین انتخابی ممکن است یا خیر. به عبارت دیگر آیا می‌توان دو اندیس **متفاوت** مانند i و j پیدا کرد $(1 \le i, j \le n)$ به طوری که شرط $a_i \cdot a_j = x$ برقرار باشد. اگر چندین جواب وجود دارد، یکی از آنها را به دلخواه برگردانید و اگر جوابی وجود ندارد `null` برگردانید.

محدودیت‌ها و ورودی

در خط اول ورودی به ترتیب دو عدد n و x آمده اند. در خط دوم n عدد آمده است که اعضای آرایه a را مشخص می‌کنند.
راهنمایی: سوال در اردر $O(n)$ با استفاده از دیکشنری قابل حل است.

- $1 \le n \le 10^5$
- $1 \le x \le 10^{18}$
- $1 \le a_i \le 10^{18}$

نمونه ۱

ورودی:

```
4 77
11 5 11 7
```

خروجی:

```
3 4
```

توضیح: در این مثال $a_3 * a_4 = 11 \cdot 7 = 77$

کلاس مسئله

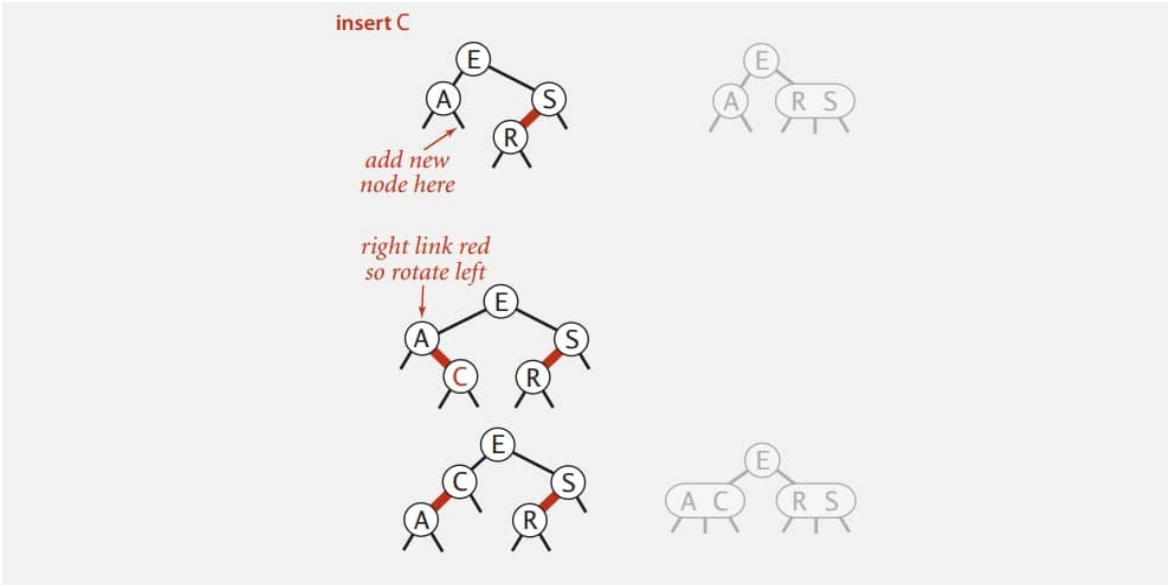
```
using System;
using TestCommon;

namespace E2
{
    public class Q2Passcode : Processor
    {
        public Q2Passcode(string testDataName) : base(testDataName)
        {
        }

        public override Action<string, string> Verifier
            => E2Verifiers.VerifyQ2Passcode;

        public override string Process(string inStr)
            => E2Processors.ProcessQ2Passcode(inStr, Solve);

        public Tuple<int,int>? Solve(long n, long x, long[] a)
        {
            throw new NotImplementedException();
        }
    }
}
```

۳ Red-Black Tree (۲۵ نمره)

در این سوال از شما می خواهیم n عدد را در قالب یک آرایه از ورودی بگیرید و با استفاده ار این اعداد یک $RedBlackTree$ بسازید. برای $pass$ کردن تست های این سوال، شما باید بعد از اضافه کردن هر عدد به این درخت، ریشه درخت را برگردانید (اعداد را به ترتیب اضافه کنید.) در ادامه به چند نکته در مورد نحوه اضافه کردن گره به $RedBlackTree$ می پردازیم.

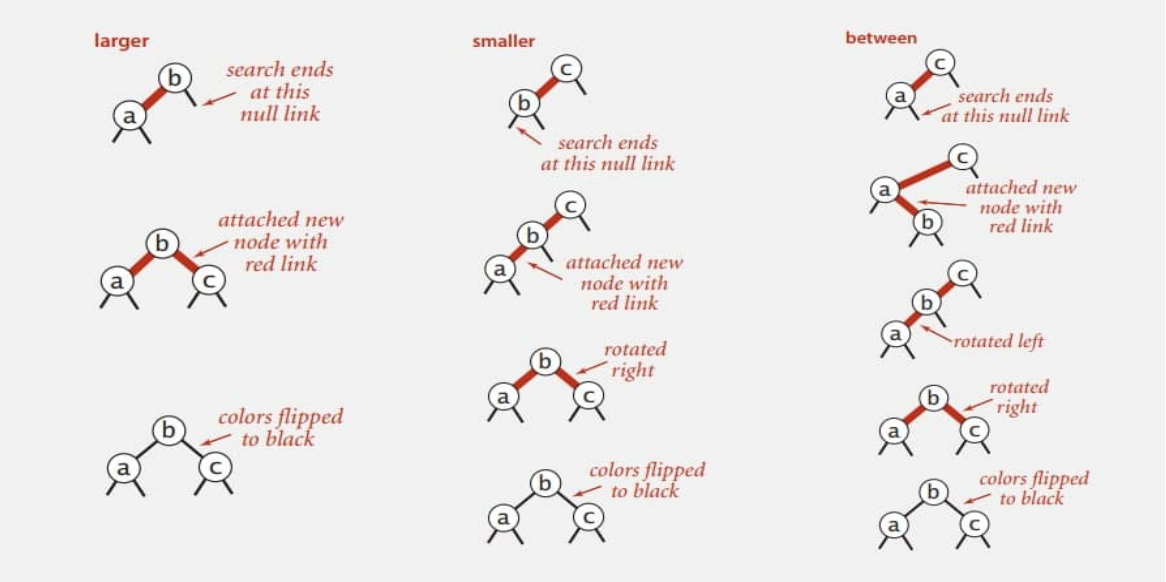
۱.۳ یال قرمز نمی تواند سمت راست باشد

شما باید در همه مراحل به این نکته توجه کنید. ممکن است این اتفاق بعد از چند بار آپدیت رنگ یال ها رخ دهد.



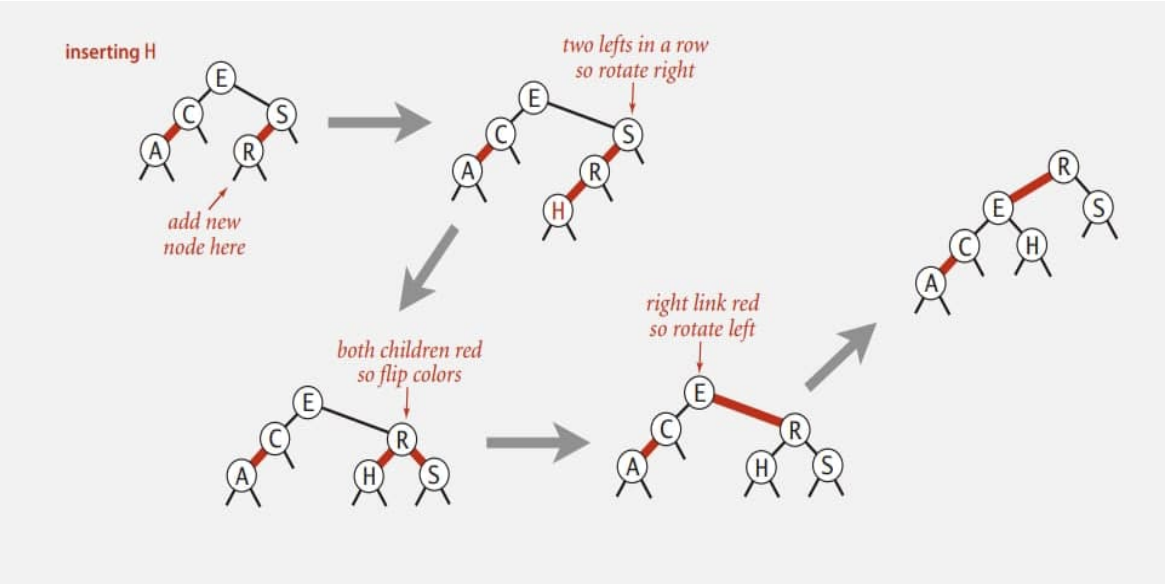
۲.۳ یال های قرمز نمی توانند پشت سر هم باشند

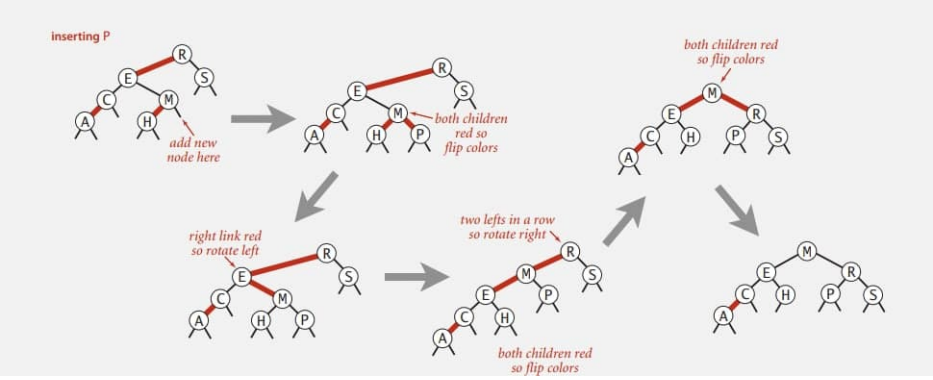
هیچ دو یال قرمزی نمی توانند گره مشترکی داشته باشند.
مثال:



۳.۳ نمونه

در نهایت چند مثال از اضافه کردن گره به $RedBlackTree$:





۴.۳ محدودیت ها

- $1 \le n \le 200$
- $1 \le a[i] \le 200$

نمونه

ورودی:

```
6
1
173
41
31
188
98
```

خروجی:

```
1 173 41 41 41 173
```

نمونه ۲

ورودی:

```
3
17
25
5
```

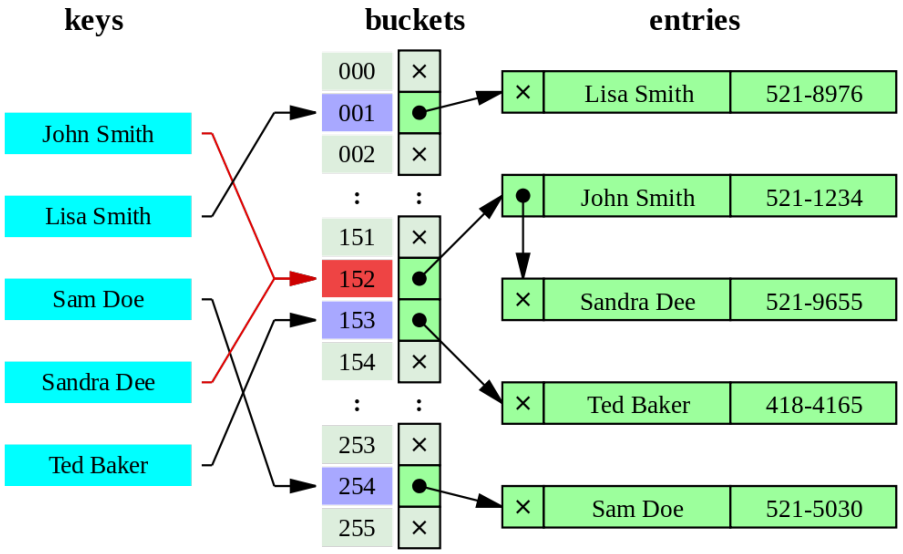
خروجی:

```
17 25 17
```

```
using TestCommon;

namespace E2
{
    public class Q3RedBlackTree : Processor
    {
        public Q3RedBlackTree(string testDataName) : base(testDataName)
        {
        }

        public override string Process(string inStr)
            => E2Processors.ProcessQ3RedBlackTree(inStr, Solve);

        public List<long> Solve(long n, long[] arr)
        {
            throw new NotImplementedException();
        }
    }
}
```

۴ Chaining Profiler (۲۵ نمره)

روش زنجیرسازی جداگانه[3] یکی از روش‌های برطرف کردن تصادم[4] در جدول هش است که در درس با آن آشنا شدید. در این روش یک کلید همواره در همان سبدی[5] که به آن هش شده است ذخیره می‌شود. با توجه به اینکه کلیدهای مختلف می‌توانند به سبدی یکسان هش بشوند، از یک ساختار داده مانند لیست پیوندی برای ذخیره کلیدهای مربوط به هر سبد استفاده می‌شود. به عنوان نمونه در شکل ۱ نحوه کارکرد روش زنجیرسازی جداگانه برای ذخیره اطلاعات چند فرد مشاهده می‌شود.



شکل ۱: مثالی از زنجیرسازی جداگانه

یکی از معایب این روش بلند شدن بیش از حد طول زنجیره‌ها است زیرا با بلندتر شدن طول یک زنجیره، زمان اجرای عملیات جستجو بر روی آن زنجیره نیز به صورت خطی افزایش می‌یابد. بنابراین توزیع یکنواخت طول زنجیره‌ها در این روش می‌تواند یک ویژگی مهم در طراحی توابع هش می‌باشد.

در این سوال به شما n رشته مانند $s_1, s_2, \cdots, s_n$ به عنوان ورودی داده می‌شود. از شما خواسته شده که این رشته‌ها را در جدول هش با روش زنجیرسازی جداگانه ذخیره کنید و علاوه بر آن واریانس تصحیح شده طول زنجیره‌ها را نیز گزارش دهید. در این سوال فرض می‌شود که از تابع هش پیش فرض زیر استفاده می‌شود که کد آن در فایل کلاس سوال موجود است:

```
۱ static int GetFNV1aHashCode(string str, int bucketCount)
۲ {
۳     if (str == null) return 0;
۴     var length = str.Length, hash = length;
۵
۶     for (int i = 0; i != length; ++i)
۷         hash = (hash ^ str[i]) * 16777619;
۸     return (hash % bucketCount + bucketCount) % bucketCount;
۹ }
```

[3] Separate Chaining
[4] Collision
[5] Bucket

توضیح: در تابع فوق پارامتر bucketCount نشان‌دهنده تعداد سبدها است. این پارامتر به عنوان ورودی در سوال داده می‌شود. برای مثال در شکل ۱ مقدار این پارامتر برابر با ۲۵۶ تنظیم شده است.
یادآوری: واریانس تحصیح شده n عدد مانند $x_1, x_2, \cdots, x_n$ به صورت زیر محاسبه می‌شود:

$$S_{n-1}^2 = \frac{\sum (x_i - \bar{x})^2}{n-1}$$

محدودیت‌ها و ورودی

در خط اول ورودی عدد n و bucketCount به ترتیب آمده اند. در هر یک از n خط بعدی یک رشته مانند s_i آمده است.

- $1 \le n \le 10^4$
- $1 \le \texttt{bucketCount} \le 10^4$
- $1 \le |s_i| \le 50$

خروجی

در خروجی یک Tuple باید برگردانید، به طوری که آیتم اول واریانس تحصیح شده طول زنجیره‌ها باشد و آیتم دوم جدول هش با روش زنجیرسازی جداگانه باشد. آیتم دوم یک لیست به طول bucketCount است که هر عضو این لیست یک لینکدلیست از رشته‌ها می‌باشد.

راهنمایی: برای انجام این سوال متد Size را در کلاس LinkedList پیاده‌سازی کنید.

نمونه ۱

ورودی:

```
5 2
Seoul
London
Buenos Aires
Lima
Guangzhou
```

خروجی:

```
0.500000
0 : London -> Buenos Aires
1 : Seoul -> Lima -> Guangzhou
```

توضیح: در این مثال واریانس برابر با $0.5 = \frac{0.5*0.5+0.5*0.5}{2-1}$ است.

کلاس مسئله

```
using System;
using TestCommon;

namespace E2
{
    public class Q4ChainingProfiler : Processor
    {
        public Q4ChainingProfiler(string testDataName) : base(testDataName)
        {
        }

        /// <summary>
        /// FNV-1a - (Fowler/Noll/Vo) is a fast,
        /// consistent, non-cryptographic hash algorithm with good dispersion.
        /// (see http://isthe.com/chongo/tech/comp/fnv/#FNV-1a)
        /// </summary>
        private static int GetFNV1aHashCode(string str, int bucketCount)
        {
            if (str == null)
                return 0;
            var length = str.Length;
            int hash = length;
            for (int i = 0; i != length; ++i)
                hash = (hash ^ str[i]) * 16777619;
            return (hash % bucketCount + bucketCount) % bucketCount;
        }

        public override string Process(string inStr)
            => E2Processors.ProcessQ4ChainingProfiler(inStr, Solve);

        // Returns:
        //      A Tuple:
        //          Item1 = Adjusted sample variance of the chain lengths
        //          Item2 = Hash table, a list of length bucketCount
        public Tuple<double, List<LinkedList<string>>> Solve(
            int n,
            int bucketCount,
            string[] s)
        {
            throw new NotImplementedException();
        }
    }
}
```

۵ Portals (۱۵ + ۲۰ نمره)

در یک بازی کامپیوتری نقشه بازی به صورت یک جدول دو بعدی با تعداد n سطر و m ستون است. دو ربات با نام‌های آلیس و باب در این جدول حضور دارند. به طور دقیق‌تر آلیس در خانه (a_{row}, a_{col}) و باب در خانه (b_{row}, b_{col}) قرار دارد. هر کدام از خانه‌های نقشه این بازی یا راهرو خالی است که با کاراکتر "." (نقطه) در جدول نمایش داده می‌شود و یا دیوار است که با کاراکتر "#" نمایش داده می‌شود. علاوه بر این دو راهرو مخصوص نیز وجود دارد که آلیس و باب در ابتدا در آنها قرار دارند و در جدول به ترتیب با کاراکتر "A" و "B" نمایش داده شده اند.
ربات‌های این بازی مجازند تا از هر راهرو ای به هر راهرو مجاور دیگری حرکت کنند. دو راهرو مجاورند اگر در یک ضلع با یکدیگر اشتراک داشته باشند.

#	#	#	#	#	#	#	#	#	#
#	.	.	.	.	A	.	.	.	#
#	.	.	#	.	.	#	#	#	#
#	#	.	#	.	.	#	.	.	#
#	.	.	#	#	#	#	B	.	#
#	.	.	#	.	.	.	.	.	#
#	#	#	#	#	#	#	#	#	#

شکل ۱: یک نمونه از جدول بازی

در طول روند بازی تعدادی "پورتال" به محیط بازی اضافه می‌شود. یک پورتال به صورت یک چهارتایی مانند (r_1, c_1, r_2, c_2) تعریف می‌شود و به این گونه عمل می‌کند که دو راهروی (r_1, c_1) و (r_2, c_2) را به طور مستقیم به هم وصل می‌کند. بنابراین هر رباتی که در خانه (r_1, c_1) باشد می‌تواند به طور مستقیم به خانه (r_2, c_2) تله‌پورت کند و بالعکس. در مجموع، در طول روند بازی k پورتال مختلف به محیط بازی اضافه می‌شود.

از شما خواسته شده تا دو زیرمسئله زیر را حل کنید.

۱.۵ زیرمسئله اول (آسان، ۲۰ نمره)

برنامه‌ای بنویسید که بررسی کند آیا در ابتدای بازی قبل از اینکه اولین پورتال به محیط بازی اضافه بشود مسیری بین آلیس و باب وجود دارد یا خیر.

۲.۵ زیرمسئله دوم (پیچیده، ۱۵ نمره)

برنامه‌ای بنویسید که بگوید پس از ظاهر شدن چندمین پورتال برای نخستین بار مسیری بین آلیس و باب به وجود می‌آید. در صورتی که در ابتدای بازی و قبل از ظاهر شدن اولین پورتال مسیر وجود دارد برنامه شما باید صفر برگرداند و اگر با وجود ظاهر شدن تمامی پورتال‌ها باز هم مسیری وجود نداشته باشد باید 1- برگرداند.

محدودیت‌ها و ورودی

در خط اول ورودی به ترتیب اعداد n و m و a_{row} و a_{col} و b_{row} و b_{col} آمده اند.
سپس در خطوط بعدی طرح نقشه بازی به صورت جدولی $n \cdot m$ آمده است.
در خط بعدی k که نمایانگر تعداد پورتال‌ها است آمده است و در هر یک از خطوط بعدی مشخصات یک پورتال آمده است. **دقت کنید که تمام اندیس‌ها در این مسئله از صفر شروع می‌شوند.**

- $1 \le n, m \le 10^3$
- $1 \le n * m \le 5 * 10^4$
- $1 \le k \le 5 * 10^4$
- $0 \le a_{row}, b_{row} < n$
- $0 \le a_{col}, b_{col} < m$

خروجی

در خروجی زیرمسئله اول فقط یک boolean باید برگردانید که مقدار آن در صورتی که در ابتدا مسیری بین آلیس و باب وجود داشته باشد برابر با `true` است و در غیر این صورت `false`.
در خروجی زیرمسئله دوم یک int باید برگردانید که مشخص می‌کند پس از ظاهر شدن چندمین پورتال برای نخستین بار مسیری بین آلیس و باب به وجود می‌آید.

نمونه ۱

ورودی:

```
7 10 3 5 5 1
##########
#.....#..#
#..##.#..#
#..#.A...#
##########
#B......##
##########
2
(2,7) <=> (3,2)
(1,5) <=> (5,2)
```

خروجی (زیر مسئله اول):

```
false
```

خروجی (زیر مسئله دوم):

```
2
```

کلاس مسئله

```
using E2.Helper;
using TestCommon;

namespace E2
{
    public class Q6PortalsB : Processor
    {
        public Q6PortalsB(string testDataName) : base(testDataName)
        {
        }

        public override string Process(string inStr)
            => E2Processors.ProcessQ6PortalsB(inStr, Solve);

        public int Solve(int n,
                         int m,
                         int a_row,
                         int a_col,
                         int b_row,
                         int b_col,
                         char[,] board,
                         List<Portal> portals)
        {
            throw new NotImplementedException();
        }
    }
}
```