



دانشگاه خواجه نصیرالدین طوسی

دانشکده مهندسی کامپیوتر

طراحی الگوریتم ها

دکتر کوهزادی

تمرین سری پنجم

نیم سال دوم 03-04



1- تبدیل واژه‌ها با کمترین تغییرات

فرض کنید دو واژه‌ی انگلیسی به نامهای `beginWord` و `endWord` به همراه یک فرهنگ لغت (`wordList`) از واژه‌های معتبر در اختیار داریم. هدف این است که تمام دنباله‌های تبدیلی ممکن را که با `beginWord` آغاز و به `endWord` ختم می‌شوند، و در هر گام فقط یک حرف تغییر کرده و واژه‌ی حاصل نیز در `wordList` وجود داشته باشد، بیابیم.

شرایط لازم برای یک دنباله‌ی تبدیلی:

- ✓ هر دو واژه‌ی متوالی در دنباله باید فقط در یک حرف تفاوت داشته باشند.
- ✓ تمامی واژه‌های میانی (s_1 تا s_k) باید در `wordList` موجود باشند. (توجه: الزامی نیست که در لیست باشد).
- ✓ واژه‌ی پایانی (s_k) باید دقیقاً برابر با `endWord` باشد.
- ✓ فقط دنباله‌ای که کمترین طول ممکن را دارند (از نظر تعداد تبدیل‌ها)، معتبر هستند.

هدف:

تمام دنباله‌های تبدیلی کوتاه‌ترین مسیر از `beginWord` به `endWord` را پیدا کنید و به صورت لیستی از واژه‌ها برگردانید. اگر هیچ دنباله‌ای وجود نداشت، خروجی باید یک لیست تهی باشد.

ورودی:

- یک رشته از حروف کوچک انگلیسی (۱ تا ۵ حرف). `beginWord`
- یک رشته به همان طول که با `beginWord` متفاوت است. `endWord`
- فهرستی شامل ۱ تا ۵۰۰ واژه‌ی منحصر به فرد، هم‌طول با `wordList`.

خروجی:

- مجموعه‌ای از لیست‌ها، که هر لیست نشان‌دهنده‌ی یک دنباله‌ی تبدیلی معتبر از `beginWord` به `endWord` با کمترین تعداد تغییرات ممکن است.
- اگر تبدیل ممکن نبود (مثلاً `endWord` در لیست نبود)، خروجی یک لیست تهی `[]` باشد.



نمونه تستها:

مثال ۱:

ورودی:

```
beginWord = "hit"
```

```
endWord = "cog"
```

```
wordList = ["hot", "dot", "dog", "lot", "log", "cog"]
```

خروجی:

```
[
```

```
["hit", "hot", "dot", "dog", "cog"],
```

```
["hit", "hot", "lot", "log", "cog"]
```

```
]
```

توضیح:

دو مسیر کوتاه و معتبر برای تبدیل وجود دارد:

hit → hot → dot → dog → cog •

hit → hot → lot → log → cog •



طراحی الگوریتم ها

مثال ۲:

ورودی:

`beginWord = "hit"`

`endWord = "cog"`

`wordList = ["hot", "dot", "dog", "lot", "log"]`

خروجی:

`[]`

توضیح:

از آنجایی که واژه‌ی پایانی `cog` در لیست وجود ندارد، هیچ دنباله‌ی تبدیلی معتبر وجود ندارد.

محدودیت‌ها:

- $1 \leq \text{beginWord.length} \leq 5$
- $1 \leq \text{wordList.length} \leq 500$
- دقيقاً هم طول با `beginWord` و `endWord` است.
- تمامی واژه‌ها فقط شامل حروف کوچک انگلیسی هستند.
- هیچ دو واژه‌ای در لیست مشابه نیستند.
- و `endWord` با هم برابر نیستند.
- مجموع طول کل دنباله‌های خروجی از 10^5 تجاوز نمی‌کند.



2- جستجوی واژه‌ها در جدول کاراکترها

فرض کنید جدولی دو بعدی به ابعاد $n \times m$ از حروف کوچک انگلیسی در اختیار داریم، به همراه یک فهرست از واژه‌ها به نام words. این است که تمام واژه‌هایی که می‌توان آن‌ها از داخل جدول پیدا کرد، استخراج کنیم.

تعریف مسیر معتبر:

هر واژه باید با کنار هم قرار دادن حروف جدول ساخته شود، به طوری که:

- ✓ حروف تشکیل‌دهندهٔ واژه باید به صورت پیوسته و مجاور در جدول قرار گرفته باشند.
- ✓ سلول‌های مجاور فقط در چهار جهت بالا، پایین، چپ و راست مجاز هستند (نه قطری).
- ✓ هر سلول جدول فقط یکبار در هر واژه قابل استفاده است.

هدف:

بررسی کنید کدام واژه‌های موجود در لیست words را می‌توان از روی جدول استخراج کرد. نتیجه باید شامل تمام واژه‌هایی باشد که مسیر معتبر برای آن‌ها وجود دارد.

وروودی:

- یک ماتریس دو بعدی از حروف کوچک انگلیسی، با ابعاد $m \times n$.
- فهرستی از واژه‌های منحصر به فرد، با حروف کوچک انگلیسی.

خروجی:

- لیستی از واژه‌هایی از words که می‌توان آن‌ها را در board یافت، به صورت مسیرهای معتبر مطابق تعریف.



نمونه تستها:

مثال ۱:

ورودی:

o	a	a	n
e	t	a	e
i	h	k	r
i	f	l	v

```
board = [
```

```
    ["o", "a", "a", "n"],
```

```
    ["e", "t", "a", "e"],
```

```
    ["i", "h", "k", "r"],
```

```
    ["i", "f", "l", "v"]
```

```
]
```

```
words = ["oath", "pea", "eat", "rain"]
```

خروجی:

```
["eat", "oath"]
```

توضیح:

فقط واژه های eat و oath مسیر معتبری در جدول دارند.



طراحی الگوریتم ها

مثال ۲:

وروودی:

a	b
c	d

```
board = [
```

```
  ["a", "b"],
```

```
  ["c", "d"]
```

```
]
```

```
words = ["abcb"]
```

خروجی:

```
[]
```

توضیح:

واژه‌ی **abcb** در جدول قابل ساخت نیست زیرا نیاز دارد که یک سلول دوبار استفاده شود که مجاز نیست.



محدودیت‌ها:

- $m == \text{board.length}$
- $n == \text{board}[i].length$
- $1 \leq m, n \leq 12$
- $\text{board}[i][j]$ is a lowercase English letter.
- $1 \leq \text{words.length} \leq 3 * 10^4$
- $1 \leq \text{words}[i].length \leq 10$
- $\text{words}[i]$ consists of lowercase English letters.
- All the strings of words are unique.

3-بیشترین تعداد ماهی قابل صید در یک شبکه

یک ماتریس دو بعدی به نام grid با ابعاد $n \times m$ به شما داده شده است. هر خانه‌ی این ماتریس $\text{grid}[r][c]$ نشان‌دهنده یکی از موارد زیر است:

- اگر مقدار آن 0 باشد، نشان‌دهنده‌ی خشکی است.
- اگر مقدار آن بزرگ‌تر از صفر باشد، نشان‌دهنده‌ی سلولی آبی حاوی تعدادی ماهی است که مقدار عددی همان تعداد ماهی را نشان می‌دهد.

ماهی‌گیر می‌تواند از هر خانه‌ی آبی دلخواهی شروع کند و می‌تواند عملیات زیر را به تعداد نامحدود انجام دهد:

1. تمام ماهی‌های موجود در سلول فعلی را صید کند.
2. به یکی از سلول‌های مجاور که آبی است حرکت کند.

مجاور بودن به معنای یکی از چهار جهت زیر است:

- بالا $(r - 1, c)$
- پایین $(r + 1, c)$
- چپ $(r, c - 1)$
- راست $(r, c + 1)$ (البته در صورتی که آن سلول در محدوده‌ی ماتریس باشد).



هدف:

بیشترین تعداد ماهی ای که می توان با انتخاب یک نقطه شروع به صورت بهینه به دست آورد را بازگردانید. اگر هیچ سلول آبی ای در ماتریس وجود نداشت، مقدار 0 را برگردانید.

مثال 1:

ورودی:

0	2	1	0
4	0	0	3
1	0	0	4
0	3	2	0

grid = [

[0, 2, 1, 0],

[4, 0, 0, 3],

[1, 0, 0, 4],

[0, 3, 2, 0]

]

خروجی:



طراحی الگوریتم ها

توضیح:

ماهی‌گیر می‌تواند از خانه‌ی (1,3) شروع کند (۳ ماهی)، سپس به (2,3) برود و ۴ ماهی دیگر صید کند.

$$\text{جمع کل ماهی‌ها} = ۳ + ۴ = ۷$$

مثال ۲:

ورودی:

1	0	0	0
0	0	0	0
0	0	0	0
0	0	0	1

grid = [

[1, 0, 0, 0],

[0, 0, 0, 0],

[0, 0, 0, 0],

[0, 0, 0, 1]

]

خروجی:



توضیح:

ماهی‌گیر می‌تواند یا از (0,0) یا (3,3) شروع کند و فقط ۱ ماهی صید کند.

Constraints:

- $m == \text{grid.length}$
- $n == \text{grid}[i].length$
- $1 \leq m, n \leq 10$
- $0 \leq \text{grid}[i][j] \leq 10$