دانشکده مهندسی کامپیوتر

جزوه درس

ساختمان‌های داده

استاد درس: سید صالح اعتمادی

پاییز ۱۳۹۸

جلسه ۱۳

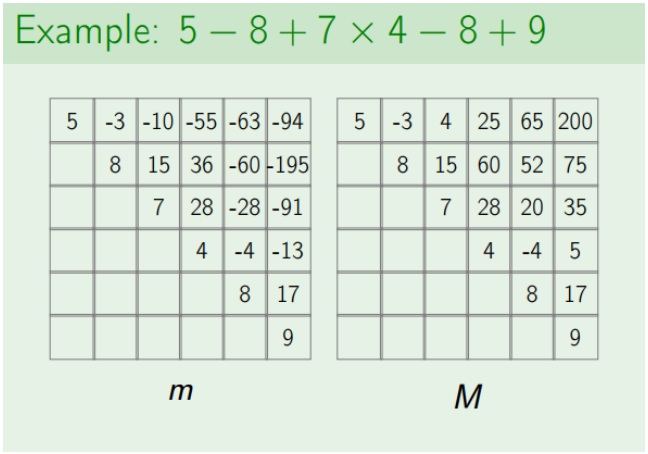
برنامه نویسی پویا — ساختار داده ای آرایه

غزل زمانی نژاد - ۱۳۹۸/۸/۱۱

جزوه جلسه ۱۳ام مورخ ۱۳۹۸/۸/۱۱درس ساختمان‌های داده تهیه شده توسط غزل زمانی نژاد.

۱.۱۳ مسئله ی پیدا کردن بیشترین مقدار یک عبارت ریاضی با پرانتزگذاری:

اصل ایده ی این مسئله روش تقسیم و حل است اما اگر به روش تقسیم و حل پیاده سازی شود، برخی از مقادیر چندین بار محاسبه شده اند. پس بهتر است به کمک برنامه نویسی پویا پیاده سازی شود.
روش حل: ابتدا دو ماتریس n^*n ، یکی برای بیشترین مقادیر در یک محدوده و دیگری برای کمترین مقادیر تشکیل می دهیم. سپس همه ی حالات پرانتزگذاری را امتحان می کنیم و دو ماتریس را با بیشترین و کمترین مقادیر پر می کنیم. بدین صورت که:
قطر اصلی هر دو ماتریس را با همان اعداد پرمی کنیم. سپس برای پر کردن سایر خانه ها، با توجه به شماره ی آن خانه، حالات مختلف پرانتزگذاری پیش می آید که هر یک از حالت ها می تواند ۴ مقدار داشته باشد. به دلیل تقارن هر یک از این دو ماتریس تنها پر کردن قطر اصلی و خانه های بالای آن کافی است. برای توضیح بیشتر چگونگی پر کردن ماتریس ها، به بررسی مثال زیر می پردازیم:



شکل ۱.۱۳: مثال نحوه پر کردن ماتریس ها

به طور مثال برای پر کردن خانه (2,5) ، ۳ حالت پرانتزگذاری زیر پیش می آید:

1. (2) + (3,4,5)

```
• Max (2,2) + Max (3,5) = 8 + 20
• Max (2,2) + Min (3,5) = 8 + -28
• Min (2,2) + Max (3,5) = 8 + 20
• Min (2,2) + Min (3,5) = 8 + -28
```

2. (2,3) * (4,5)

```
• Max (2,3) * Max (4,5) = 15 * -4
• Max (2,3) * Min (4,5) = 15 * -4
• Min (2,3) * Max (4,5) = 15 * -4
• Min (2,3) * Min (4,5) = 15 * -4
```

3. (2,3,4) – (5)

```
• Max (2,4) - Max (5,5) = 60 - 8
• Max (2,4) - Min (5,5) = 60 - 8
• Min (2,4) - Max (5,5) = 36 - 8
• Min (2,4) - Min (5,5) = 36 - 8
```

در نهایت برای پر کردن خانه $(2,5)$ ماتریس مقادیر مینیمم، کمترین مقدار ۱۲ حالت بالا، و برای پر کردن خانه $(2,5)$ ماتریس مقادیر ماکسیمم، بیشترین مقدار حالات بالا را در نظر می گیریم.
شبه کد این مسئله به صورت زیر است:

Algorithm 1 MinAndMax(i, j)

```
1: min ← +∞
2: max ← −∞
3: for k from i to j − 1 do
4:     a ← M(i, k) op_k M(k + 1, j)
5:     b ← M(i, k) op_k m(k + 1, j)
6:     c ← m(i, k) op_k M(k + 1, j)
7:     d ← m(i, k) op_k m(k + 1, j)
8:     min ← min(min, a, b, c, d)
9:     max ← max(max, a, b, c, d)
10: end for
11: return (min, max)
```

Algorithm 2 Parentheses($d_1\ op_1\ d_2\ op_2\ \ldots\ d_n$)

```
1: for i from 1 to n do
2:     m(i, i) ← d_i, M(i, i) ← d_i
3: end for
4: for s from 1 to n − 1 do
5:     for i from 1 to n − s do
6:         j ← i + s
7:         m(i, j), M(i, j) ← MinAndMax(i, j)
8:     end for
9: end for
10: return M(1, n)
```

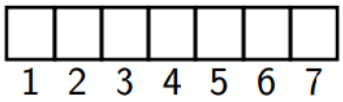
پیچیدگی زمانی الگوریتم ۱، حداکثر n است.
پیچیدگی زمانی الگوریتم ۲، حداکثر n^3 است.

۲.۱۳ ساختار داده: آرایه

آرایه بخشی از حافظه اصلی است که پیوسته بوده (یعنی عناصر آن در حافظه پشت هم قرار می گیرند.) و دارای اندازه مشخص می باشد. دسترسی به هر عنصر آرایه از $O(1)$ است. چون با pointer arithmetic میتوان محل هر عنصر را محاسبه کرد.

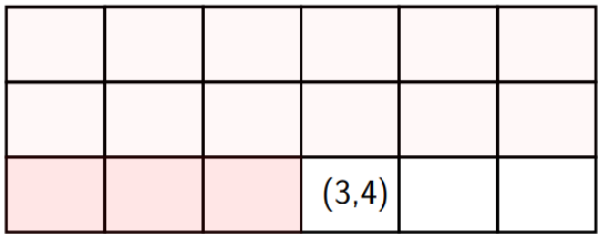
Constant-time access

$$\text{array\_addr} + \text{elem\_size} \times (i - \text{first\_index})$$



شکل ۲.۱۳: دسترسی به عناصر در آرایه یک بعدی

Multi-Dimensional Arrays

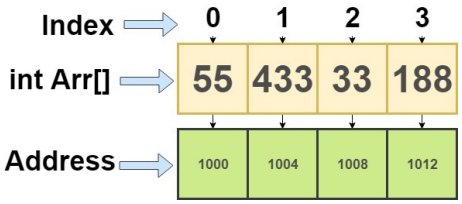


$$\text{array\_addr} +$$
$$\text{elem\_size} \times ((3-1) \times 6 + (4-1))$$

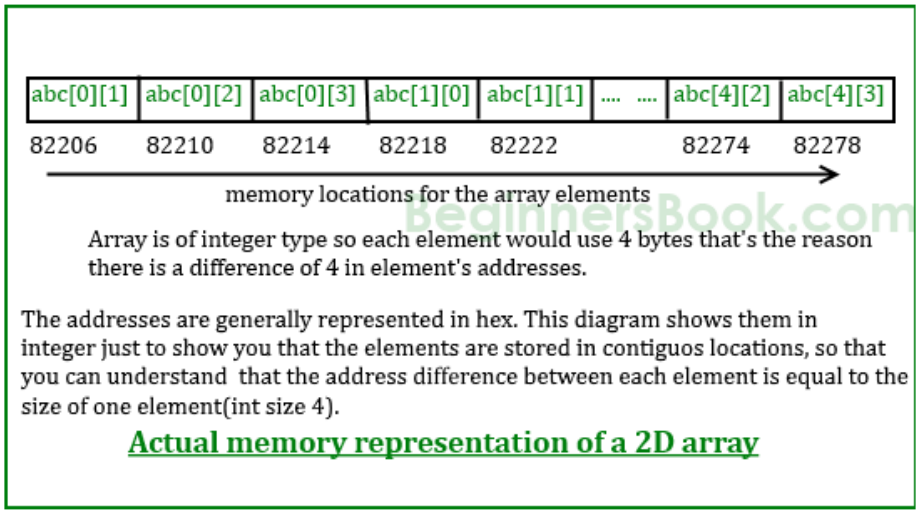
شکل ۳.۱۳: دسترسی به عناصر در آرایه چند بعدی

برای محاسبه مقدار حافظه ای که هر آرایه اشغال می کند، باید سایز آرایه را در مقدار حافظه ای که هر المان آرایه به خود اختصاص می دهد ضرب کنیم.
به طور مثال، حافظه ای که $\text{int array}[100]$ اشغال می کند، برابر است با:

$$100 * \text{sizeof(int)} = 400$$



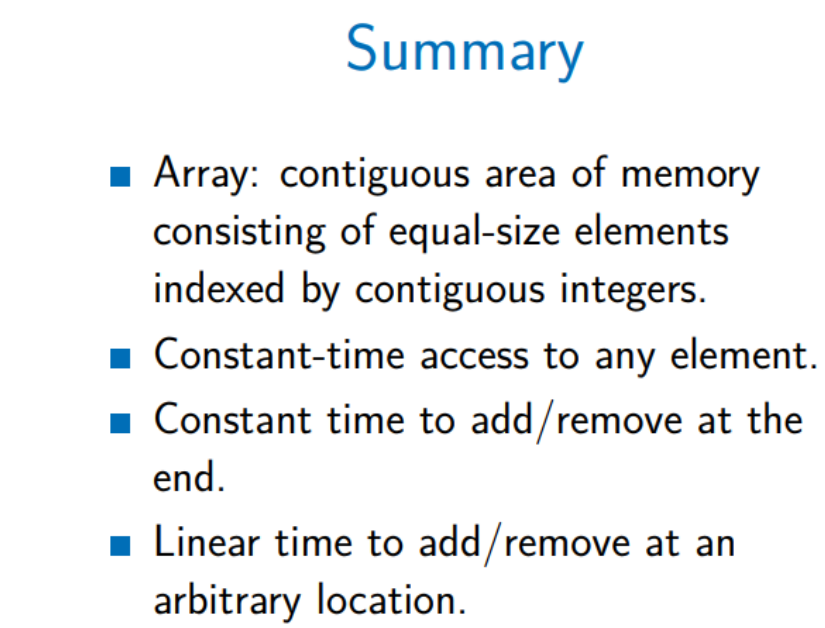
شکل ۴.۱۳: عناصر یک آرایه یک بعدی به همراه آدرس هایشان
[۱]



شکل ۵.۱۳: عناصر یک آرایه چند بعدی به همراه آدرس هایشان
[۲]

۳.۱۳ عملیات اضافه یا حذف کردن یک عنصر در آرایه:

- پیچیدگی اضافه کردن عنصر به انتهای آرایه $O(1)$ است چون اگر در انتهای آرایه جای خالی داشته باشیم، می توان یک عنصر اضافه کرد. پاک کردن از انتها نیز به همین شکل است.
- پیچیدگی اضافه کردن عنصر به ابتدای آرایه از $O(n)$ است چون باید همه عناصر را یکی به جلو shift بدهیم. برای پاک کردن از ابتدا، باید بعد از پاک کردن اولین عنصر، بقیه عناصر را یکی به عقب shift بدهیم که این کار در $O(n)$ انجام می شود.
- اضافه یا حذف کردن از وسط آرایه نیز در $O(n)$ انجام می شود.



شکل ۶.۱۳: خلاصه مبحث آرایه

Bibliography

[1] "Pointer to array." https://simplesnippets.tech/cpp-pointer-to-an-array/.

[2] "Pointers and 2d array." https://beginnersbook.com/2014/01/2d-arrays-in-c-example/.