
دانشکده مهندسی کامپیوتر

جزوه درس

# ساختمان‌های داده

استاد درس: سید صالح اعتمادی

پاییز ۱۳۹۸

# جلسه ۱

# آشنایی با الگوریتم

میلاد اسفندیاری فر - ۱۳۹۸/۷/۱

جزوه جلسه ۱ام مورخ ۱۳۹۸/۷/۱درس ساختمان‌های داده تهیه شده توسط میلاد اسفندیاری فر. در جهت مستند کردن مطالب درس ساختمان های داده، بر آن شدیم که از دانشجویان جهت مکتوب کردن مطالب کمک بگیریم. هر دانشجو می‌تواند برای مکتوب کردن یک جلسه داوطلب شده و با توجه به کیفیت جزوه از لحاظ کامل بودن مطالب، کیفیت نوشتار و استفاده از اشکال و منابع کمک آموزشی، حداکثر یک نمره مثبت از بیست نمره دریافت کند. خواهش‌مند است نام و نام خانوادگی خود، عنوان درس، شماره و تاریخ جلسه در ابتدای این فایل را با دقت پر کنید.

## ۱.۱ الگوریتم و کاربرد آن چیست؟

هر برنامه ای تشکیل شده از یک سری کارهای متوالی و پشت سرهم تا نهایتا به هدف برنامه منتهی شود. گاهی این روند ساده است مانند چاپ عدد ۱ تا ۱۰۰ اما گاهی پیچیدگی آن زیاد میشود مانند پیدا کردن یک کلمه در درون یک متن که بسته به طول کلمه و طول متن پیچیدگی آن بیشتر میشود و در این شرایط اهمیت الگوریتم و بهینه بودن آن مشخص میشود مانند سرچ گوگل که در کسری از ثانیه نتایج را پیدا میکند در صورتی که اگر یک فردی که دانش الگوریتم ندارد بخواهد این قابلیت را پیاده سازی کند، زمان بسیار بیشتری از گوگل طول بکشد. در درس ساختمان و طراحی الگوریتم نحوه کار به این شکل است که ورودی و خروجی کاملا مشخص است و صرفا هدف ما سریع انجام دادن کار ها تا رسیدن به خروجی مد نظر است. حال برای شهود بیشتر به الگوریتم و اهمیت آن به یک مثال میپردازیم.

### ۱.۱.۱ مسئله محاسبه دنباله اعداد فیبوناچی.

همه ما با دنباله زیر اشنایی داریم که به آن دنباله اعداد فیبوناچی میگوییم:

$$1,1,2,3,5,8,13,21,...$$

که فرمول کلی آن به صورت زیر است :

$$f_n = f_{n-1} + f_{n-2} \quad (f_0 = 1, f_1 = 1)$$

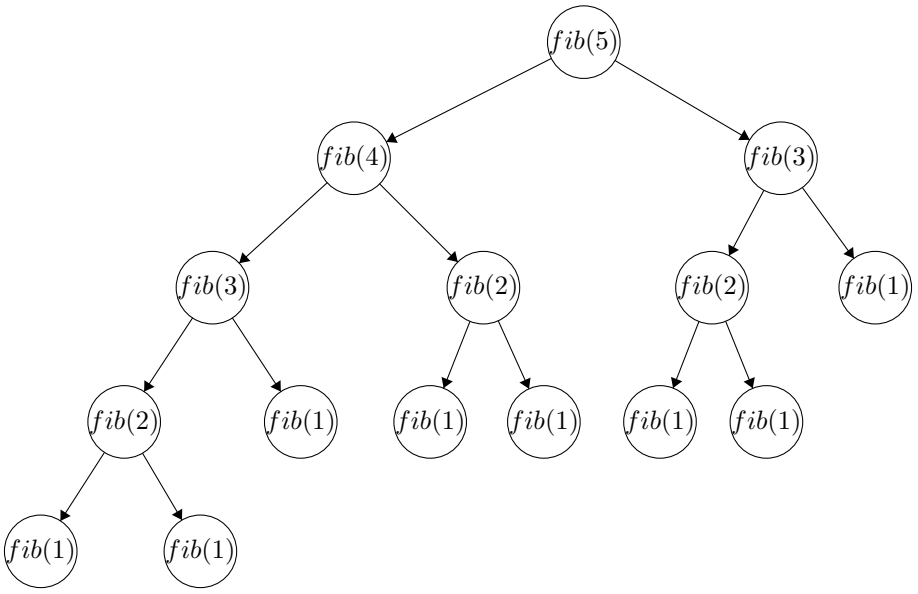
برای حل این مسئله ابتدا اولین راه حلی که به ذهنمان بعد از دیدن فرمول محاسبه دنباله فیبوناچی میرسد یعنی راه حل بازگشتی حل میکنیم سپس الگوریتم آن را تحلیل میکنیم و در آخر سعی میکنیم الگوریتم بهینه تری را جایگزین آن کنیم.

آ) **راه حل بازگشتی محاسبه جمله n ام دنباله فیبوناچی** برای حل کردن مسئله فیبوناچی به صورت بازگشتی کافیست فقط حواسمان به پایه تابع بازگشتی باشد که در حلقه بینهایت گیر نکنیم. و مقدار بازگشتی کلی تابع دقیقا مانند خود فرمول آن است. قطعه کد زیر نمونه حل آن در زبان برنامه نویسی سی شارپ است.

```
۱    private static long Fib(long n)
۲    {
۳        if(n == 0 || n == 1)
۴            return 1;
۵        else
۶            return Fib(n - 1) + Fib(n - 2);
۷    }
```

ب) **:بررسی الگوریتم بازگشتی محاسبه جمله n ام دنباله فیبوناچی** بعد از پیاده سازی کد بالا متوجه میشویم که الگوریتم برای n های بزرگ تر بسیار کند میشود تا حدی که از جمله ۵۰ ام در نظر میگیریم محاسبه نمیشود. حال علت آن را از روی درخت بازگشتی روش بالا نشان میدهیم.

شکل ۱.۱ به طور مثال درخت واره روش بازگشتی برای محاسبه جمله ۵ام دنباله فیبوناچی می باشد. باتوجه به شکل میتوان مشاهده کرد که جملات به دفعات متعدد تکراری محاسبه می شوند . به طور مثال جمله ۲ام ۳ بار حساب شده است و علت زمان بالا این روش همین محاسبات تکراری است. برای حل مشکل کافی است مسئله فیبوناچی را به روش بهینه حل کنیم.




شکل ۱.۱: درخت محاسبه جمله پنجم فیبوناچی

**ج) :روش بهینه و سریع تر مسئله فیبوناچی**

کافی است دنباله را یک آرایه n تایی در نظر بگیریم که در هر خانه مقدار n ام دنباله فیبوناچی قرار دارد. و برای مقدار دهی هر خانه کافی است در هر خانه جمع دو خانه قبلی را بریزیم فقط باید توجه داشت که دو خانه اول به مقدار ۱ مقداردهی شده باشند.

قطعه کد زیر راه حل بهینه شده برای مسئله فیبوناچی در زبان برنامه نویسی سی شارپ می باشد.

```
۱     private static long Fib(long n)
۲     {
۳         long[] fib = new long[n+1];
۴         fib[0] = 1;
۵         fib[1] = 2;
۶
۷         for(int i = 2; i <= n; i++)
۸             fib[i] = fib[i-1] + fib[i-2];
۹         return fib[n];
۱۰    }
```

با مقایسه سرعت روش بهینه با روش بازگشتی متوجه اهمیت فوق العاده الگوریتم میشویم

# Bibliography