طراحی الگوریتم ها

مقدمه

مطالب مورد بحث

- تعریف الگوریتم
- اهمیت توسعه الگوریتم های کارا
  - مثال: دنباله اعداد فیبوناتچی

تعریف الگوریتم

- به روش حل یک مسئله، الگوریتم گفته می شود.
- الگوریتم یک روال محاسباتی/عملیاتی خوش تعریف است که تعدادی (صفر یا بیشتر) مقدار بعنوان ورودی می گیرد و تعدادی (یک یا بیشتر) مقدار بعنوان خروجی تولید می نماید.
  - خروجی الگوریتم ممکن است صریح نباشد ← اثرات جانبی (side effects)
- منظور از خوش تعریف بودن روال
  - هر یک از دستورالعمل های آن باید دقیق و بدون ابهام باشد.
  - ترتیب اجرای دستورالعمل های آن کاملا مشخص باشد.
  - اجرای دستورالعمل ها باید در زمانی متناهی خاتمه یابد.

بیان الگوریتم

- الگوریتم را می توان به طرق مختلف بیان نمود.
  - به زبان طبیعی (دستورالعمل)
  - با استفاده از نمودار های خاص (فلوچارت)
  - با استفاده از زبانی شبیه یکی از زبان های برنامه نویسی (شبه کد)
  - با استفاده از یکی از زبان های برنامه نویسی (برنامه)
- در این کلاس ما از دو روش آخر استفاده می کنیم.
  - زبان C و C++

اهمیت توسعه الگوریتم های کارا

- برای هر مساله ای بیش از یک الگوریتم می توان طراحی نمود.
- ممکن است این الگوریتم ها از نظر کارآیی با هم متفاوت باشند.
- منظور از کارایی
  - زمان اجرای الگوریتم برای حل یک نمونه از مسئله
  - میزان حافظه مصرفی الگوریتم
  - هزینه نگهداری و بروز رسانی و استفاده الگوریتم (مثلا تنظیم پارامترها)
  - هزینه سخت افزاری که الگوریتم بر روی آن اجرا می شود.
  - خوانایی الگوریتم
  - امنیت الگوریتم
  - ...

اهمیت توسعه الگوریتم های کارا

- معمولا مهمترین عوامل در تعیین کارایی یک الگوریتم، زمان اجرای آن و سپس میزان مصرف حافظه آن است.

مثال: دنباله اعداد فیبوناتچی

- مطلوب است الگوریتمی که عدد n را بگیرد و fib(n) را بعنوان خروجی بازگرداند.

$$fib(n) = \begin{cases} 0 & \text{if } n = 0 \\ 1 & \text{if } n = 1 \\ fib(n-1) + fib(n-2) & \text{else} \end{cases}$$

Fibonacci sequence :

$0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, \ldots$

مثال: دنباله اعداد فیبوناتچی

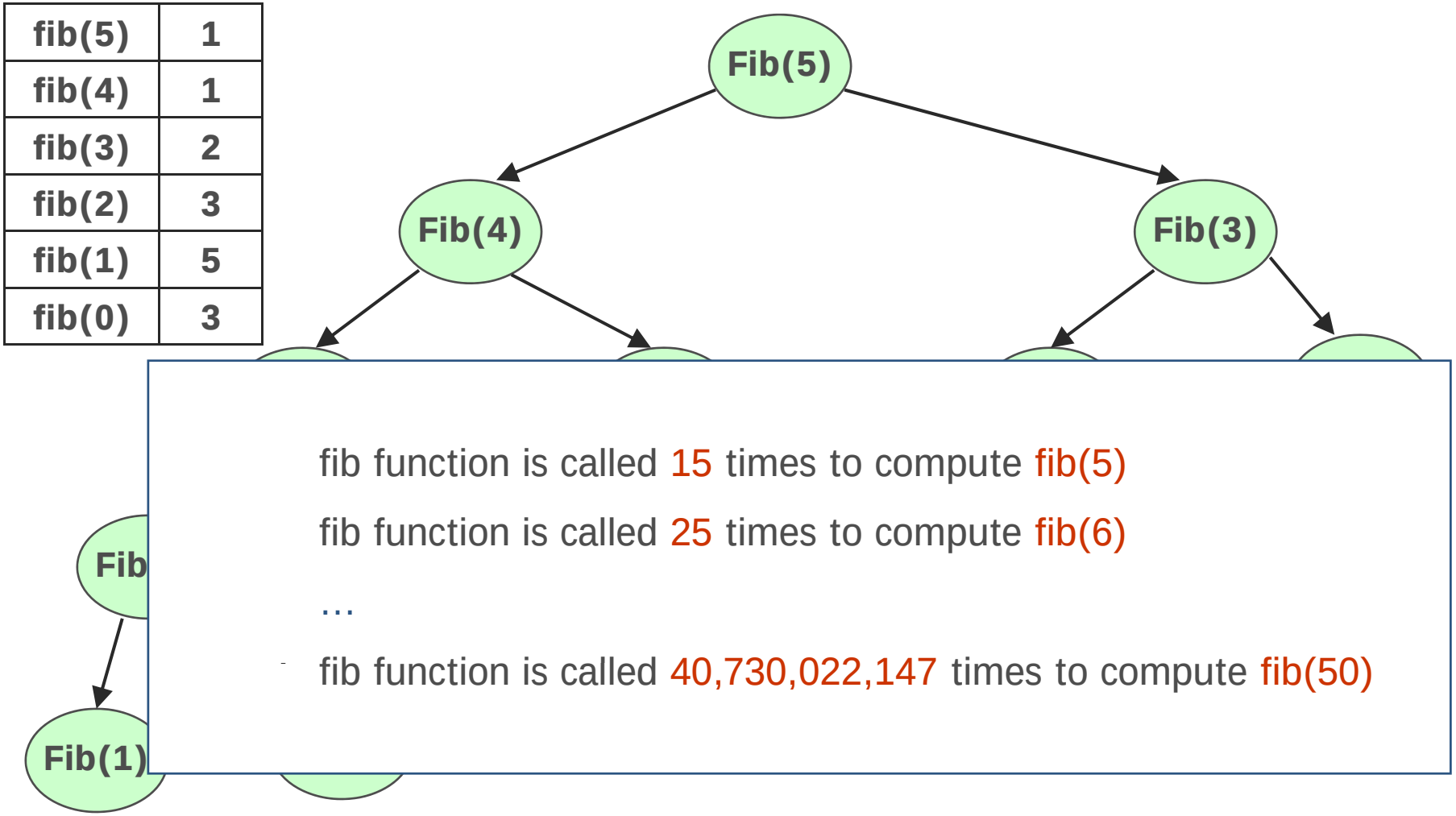
```
long int fib(int n) {
   if ( n <= 1) return n;
   else return fib(n-1)+fib(n-2);
}
```

- محاسبه fib(50) بر روی یک کامپیوتر خانگی معمولی، ۴۴۸ ثانیه یعنی حدود ۷.۵ دقیقه طول کشید.
- fib(50) = 12586269025
- چرا الگوریتم اینقدر کند است؟

نحوه محاسبه fib(5) توسط الگوریتم

تعداد تکرار محاسبه

fib(5)	1
fib(4)	1
fib(3)	2
fib(2)	3
fib(1)	5
fib(0)	3



fib function is called 15 times to compute fib(5)

fib function is called 25 times to compute fib(6)

…

fib function is called 40,730,022,147 times to compute fib(50)

مثال: دنباله اعداد فیبوناتچی

- عیب الگوریتم و منشا سرعت کم آن: محاسبه مجدد مقادیر عناصر دنباله فیبوناتچی
- راه حل: هر یک از عناصر دنباله فقط یک بار محاسبه شود.
  - از یک آرایه کمکی fibValues[] استفاده شود. که ابتدا
  - fibvalues[0] = 0;
  - fibValues[1] = 1;
  - fibValues[i] = -1; i>1
  - هر بار که نیاز به fib(i) داریم بررسی شود که اگر fibValues[i] مقدارش مخالف ۱- است از آن استفاده شود در غیر اینصورت تابع fib(i) فراخوانی شود و مقدار بازگشتی اش در fibValues[i] قرار داده شود.

مثال: دنباله اعداد فیبوناتچی

```
long int fibValues[MAX_COUNT];
long int fib(int n) {
   if ( n <= 1) return n;
   else {
      if(fibValues[n-1]== -1)
            fibValues[n-1]  = fib(n-1);
      if(fibValues[n-2]== -1)
            fibValues[n-2]  = fib(n-2);
     return fibValues[n-1]+fibValues[n-2];
   }
}
```

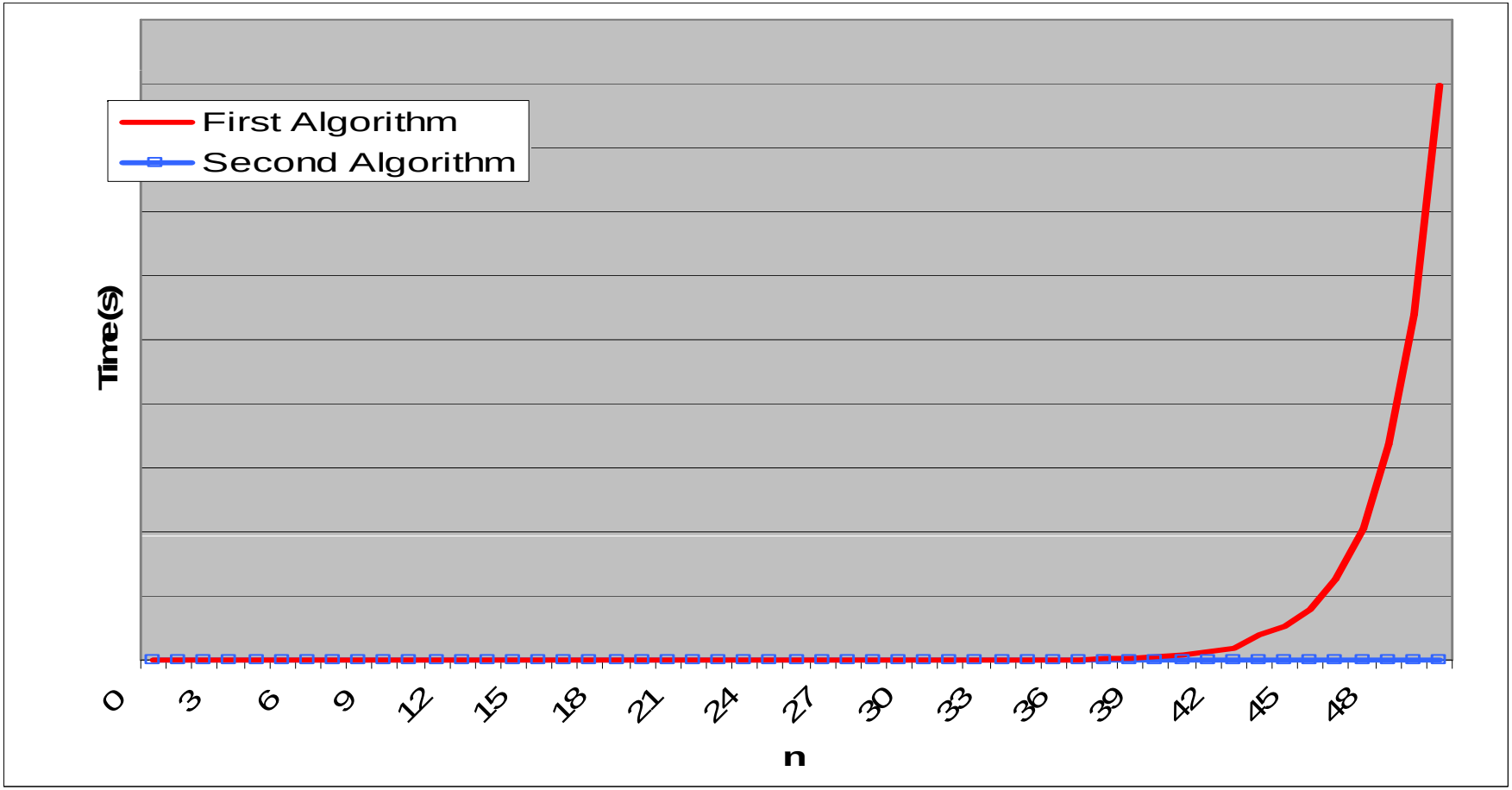
مثال: دنباله اعداد فیبوناتچی

- این راه حل، مشکل محاسبه مجدد را حل می کند.
- اما دو نکته قابل توجه است
  - ابعاد آرایه را چقدر انتخاب کنیم ← هدر رفتن فضا
  - آیا واقعا به آرایه نیاز داریم؟
- راه حل بهتر (بهبود کارایی از منظر میزان حافظه مورد استفاده)
  - نیازی به فراخوانی بازگشتی نیست ← استفاده از حلقه
  - نیازی به استفاده از آرایه نیست. ← استفاده از دو متغیر برای ذخیره آخرین دو عنصر محاسبه شده
  - 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, …

مثال: دنباله اعداد فیبوناتچی

```
long int fib(int n) {
    if ( n == 0) return 0;
    else if (n == 1) return 1;
    long int F1 = 0,     F2 = 1, result ;
    for(int i=1;i<n;i++) {
            result = F1 + F2;
            F1 = F2;
            F2 = result;
    }
    return result;
}
```

مثال: دنباله اعداد فیبوناتچی



نمودار زمان اجرای الگوریتم بازگشتی فیبوناتچی بر حسب مقدار n

نتیجه گیری

- همیشه اولین یا ساده ترین الگوریتم، بهترین الگوریتم نمی باشد.
- یک الگوریتم بد نیز ممکن است در برخی نمونه ها خوب عمل کند.
- نمی توان ابتدا الگوریتم را پیاده سازی کرد و سپس کارایی آن را ارزیابی نمود.
- برای ارزیابی کارایی الگوریتم ها نیازمند روشی تئوریک هستیم
  - که کارایی الگوریتم را پیش از پیاده سازی بتوان تحلیل نمود.
  - مستقل از مشخصات ماشینی که الگوریتم بر روی آن اجرا می شود.
  - مستقل از یک نمونه خاص باشد.
- نتیجه: باید رفتار الگوریتم را شناسایی کنیم. ← تحلیل مجانبی