تحلیل بهترین حالت (Best case analysis)

- در این حالت آرایه ای که به عنوان ورودی داده شده است خودش مرتب شده است .
- همیشه $A[i] \leq key$
- همه ی t_j ها برابر با ۱ هستند .
- زمان اجرا برابری شود با :

$$T(n) = c_1 n + c_2(n-1) + c_4(n-1) + c_5(n-1) + c_8(n-1)$$

$$= (c_1 + c_2 + c_4 + c_5 + c_8)n - (c_2 + c_4 + c_5 + c_8)$$

$$= a\,n + b$$

یک تابع خطی است از n .

تحلیل بدترین حالت (worst case analysis)

- در این حالت آرایه ی ورودی برعکس مرتب شده است .
- همیشه $A[i] > key$
- باید key با همه خانه های سمت چپ خانه ی jام مقایسه شود. ($j-1$ بار)
- هر بار که از حلقه while خارج می شویم بدلیل آنست که i صفر می شود، پس هر بار یک مرتبه بیشتر اجرا می شود.

$$\sum_{j=2}^{n} t_j = \sum_{j=2}^{n} j = \frac{n(n+1)}{2} - 1$$ ←

$$\sum_{j=2}^{n} (t_j - 1) = \sum_{j=2}^{n} (j-1)$$

$$T(n) = c_1 n + c_2(n-1) + c_4(n-1) + c_5\left(\frac{n(n+1)}{2} - 1\right) + c_6\left(\frac{n(n-1)}{2}\right) + c_7\left(\frac{n(n-1)}{2}\right) + c_8(n-1)$$

$$= a\,n^2 + b\,n + c$$

- $T(n)$ یک تابع درجه دوم از n است .

ما معمولاً تمرکز می‌کنیم که زمان اجرای بدترین حالت را بدست آوریم که در واقع بلندترین زمان اجرای به ازای هر ورودی ممکن از سایز n است، دلایل زیر وجود دارد:

- بدترین زمان اجرا یک حد بالای تضمین شده به ازای تمام ورودی‌ها می‌دهد.
- برای بسیاری از الگوریتم‌ها، معمولاً همان بدترین حالت اغلب رخ می‌دهد.
  - در الگوریتم searching، هرگاه به دنبال آیتمی بگردیم که وجود ندارد.
- اغلب تحلیل حالت میانگین به بدی تحلیل بدترین حالت است البته از لحاظ order
- order چیست؟

order of growth:

- برای بدست آوردن order یک فرمول تابع زمان اجرا:
  - جملات با ارزش کمتر را دور می‌ریزیم.
  - ضریب ثابت را حذف می‌کنیم.
  - مثال: $T(n) = 5n^3 + 4n^2 - 2n + 5 = \theta(n^3)$ می‌گوییم $T(n)$ مانند n^3 رشد می‌کند.

چرا order در تحلیل الگوریتم‌ها مهمتر از خود تابع است؟

- الگوریتمی از الگوریتم دیگر کارآمدتر است اگر order آن بهتر باشد. (کمتر باشد)
- مقایسه تابع کافی نیست.

* طراحی کردن الگوریتم‌ها:
  - مرتب سازی insertion sort به سبک طراحی incremental است.

incremental algorithm چیست ؟

- یک الگوریتم incremental یک دنباله از ورودی می گیرد .
- یک دنباله از جواب ها می یابد .
- دنباله جوابها به صورت گام به گام و افزایش به ازای در نظر گرفتن تعداد بیشتری از ورودی ها ساخته می شود

مثال : در مسئله insertion sort ورودی زیر داده شده باشد .

input sequence: 5 2 9 3 4 8

یک دنباله از جوابها به صورت زیر ساخته می شود .

sequence of solutions:

- گام اول فقط با در نظر گرفتن دو آیتم اول ورودی

2 5 ← مهم نیست ← بقیه

- گام دوم با در نظر گرفتن و اضافه کردن سومین آیتم از ورودی :

2 5 9 ← بقیه مهم نیست ←

- گام سوم ، با اضافه کردن (increment) آیتم چهارم از ورودی :

2 3 5 9 ← مهم نیست ←

- به همین روال :

- دنباله ای قابل استفاده از راه حل ها بدست می دهد ، این تکنیک پایه ای برای نوع خاصی از الگوریتم ها به نام online algorithm هست .

: online algorithm

- در این نوع الگوریتم به دلیل حجم بسیار بالای ورودی و سایز n ، یا به هر دلیل دیگر تمام ورودی یکجا و در ابتدای شروع الگوریتم در دسترس الگوریتم نیست. (در incremental در دسترس هست و ما سبک طراحی الگوریتم را خودمان به دلخواه incremental تعیین کرده ایم)
- الگوریتم به مرور با ورودی تغذیه می شود.
- الگوریتم ورودی خود را تکه به تکه براساس ترتیبی که ورودی به آن داده می شود پردازش می کند و منتظر نمی ماند که تمام ورودی را داشته باشد و بعد پردازش کند.

: offline algorithm

- تمام اطلاعات و داده‌های مسئله در همان ابتدا داده شده است.
- اکثر یا همه الگوریتم‌های پیش روی این درس offline است، مگر آنکه گفته شود.

نکته: راه‌ها و روش‌های زیادی برای طراحی الگوریتم وجود دارد که به فراخور شرایط مسئله و efficiency از بین آنها می توان انتخاب کرد.

طراحی الگوریتم (Algorithm Design) : یک روش و یا یک فرآیند ریاضی است که برای حل مسئله (problem-solving) و مهندسی الگوریتم‌ها به کار می رود.

مراحل و گام‌های توسعه الگوریتم‌ها و نقش طراحی الگوریتم:

1. Problem definition
2. Development of a model
3. Specification of the algorithm
4. Designing an algorithm
5. Checking the correctness of the algorithm
6. Analysis of algorithm
7. Implementation of algorithm
8. Program testing
9. Documentation preparation

: Designing algorithms

- یک رویکرد متداول در طراحی الگوریتم‌ها Divide and conquer است.

گام‌ها: Divide : تقسیم مسئله به یک تعداد زیرمسئله

Conquer : اگر زیرمسئله به اندازه کافی کوچک هست آن گاه آن را با روش brute force حل کن.

combine : جواب‌های زیرمسائل را ترکیب کن تا به جواب مسئله اصلی برسی.