ساختمان داده ها

کران پایین زمان مرتب سازی برای الگوریتم های مبتنی بر مقایسه

مدرس: غیاثی‌شیرازی
دانشگاه فردوسی مشهد

مرتب سازی مبتنی بر مقایسه

- می گوییم یک الگوریتم مرتب سازی مبتنی بر مقایسه است، هرگاه به هیچ طریق دیگری جز مقایسه عناصر مقدار آنها را بررسی نکند.
- در ادامه فرض می کنیم که عناصر ورودی الگوریتم مرتب سازی تکراری نیستند.
  - بنابراین فرض می کنیم که تمام مقایسه ها به فرم $a_i \leq a_j$ هستند.

مدل درخت تصمیم برای الگوریتم های مبتنی بر مقایسه

- هر الگوریتم مرتب سازی مبتنی بر مقایسه را می توان با یک درخت تصمیم نشان داد (برای n ثابت)
- درخت تصمیم یک درخت دودویی است.
- گره های میانی این درخت یک مقایسه بین دو عنصر از آرایه اولیه را نشان می دهند.
- گره های برگ، یک جایگشت از داده ها را مشخص می کنند که با آن جایگشت داده ها مرتب می شوند.
- اجرای الگوریتم مرتب سازی متناظر است با دنبال کردن یکی از مسیر های درخت تصمیم از ریشه تا یکی از برگ ها.

مدل درخت تصمیم- مثال

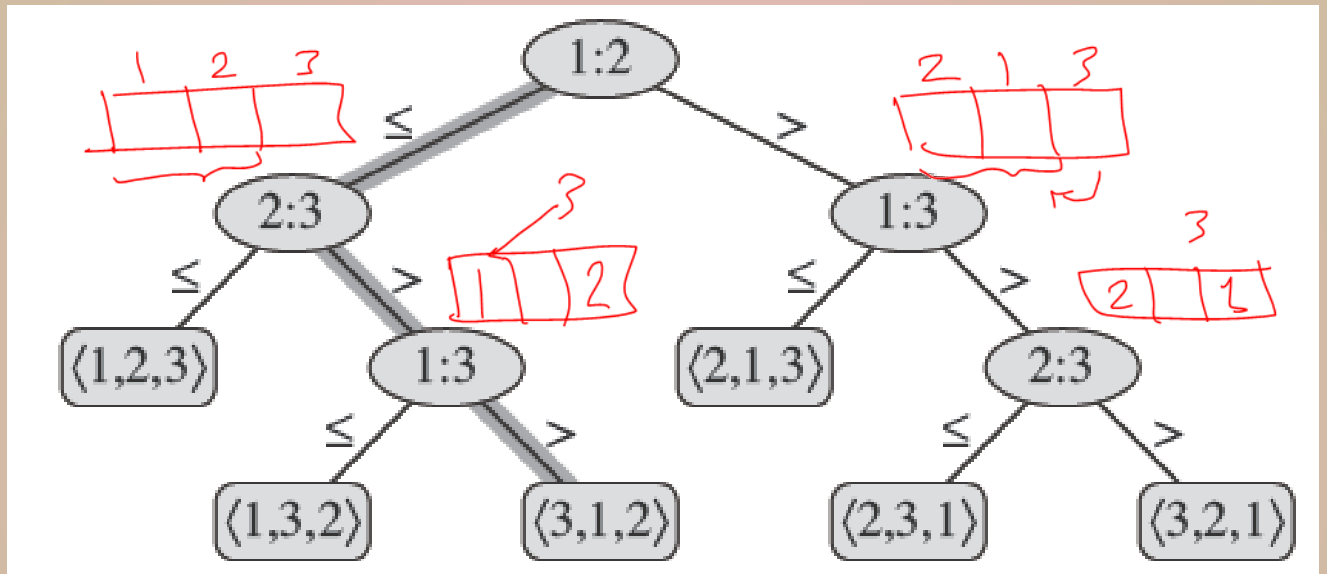
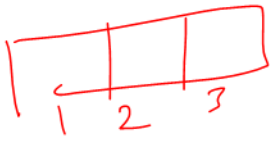
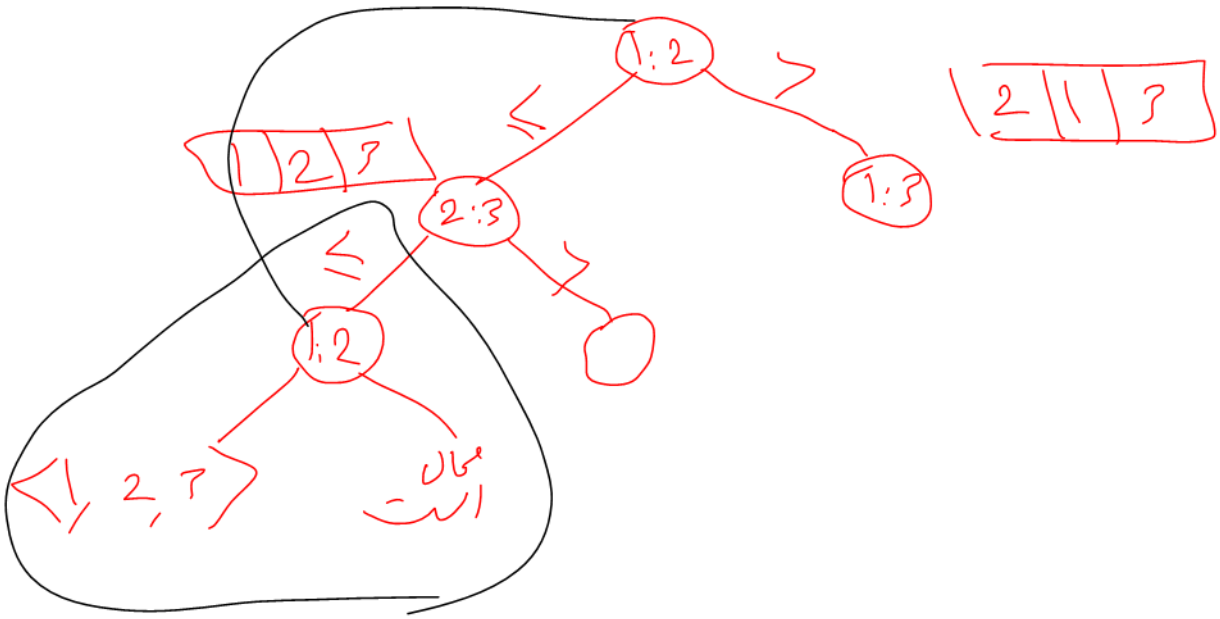


Figure 8.1 The decision tree for insertion sort operating on three elements. An internal node an notated by $i:j$ indicates a comparison between a_i and a_j. A leaf annotated by the permutation $\langle \pi(1), \pi(2), \ldots, \pi(n) \rangle$ indicates the ordering $a_{\pi(1)} \leq a_{\pi(2)} \leq \cdots \leq a_{\pi(n)}$. The shaded path indicates the decisions made when sorting the input sequence $\langle a_1 = 6, a_2 = 8, a_3 = 5 \rangle$; the permutation $\langle 3, 1, 2 \rangle$ at the leaf indicates that the sorted ordering is $a_3 = 5 \leq a_1 = 6 \leq a_2 = 8$. There are $3! = 6$ possible permutations of the input elements, and so the decision tree must have at least 6 leaves.

درخت تصمیم BubbleSort بهینه نشده بدی کار کرده





مدل درخت تصمیم- حالت عمومی

- n عنصر را می خواهیم مرتب کنیم.

$$A[1], A[2], \dots, A[n]$$

- گره داخلی به فرم i:j
  - زیردرخت چپ =>
  - زیر درخت راست<
- گره برگ یک جایگشت از n عنصر را نشان می دهد.

$$\langle \pi(1), \dots, \pi(n) \rangle$$

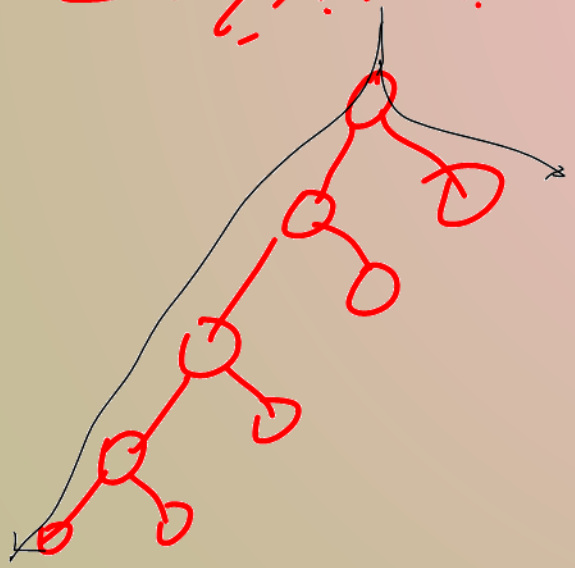
$$A[\pi(1)] \leq \cdots \leq A[\pi(n)]$$

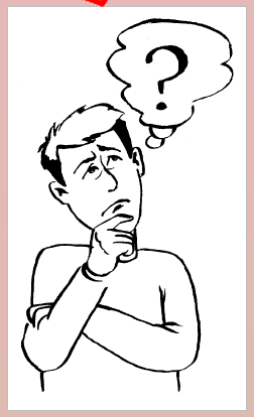
درخت تصمیم، الگوریتم های مرتب سازی قطعی مبتنی بر مقایسه را مدل می کند

- الگوریتم قطعی (Deterministic) است نه تصادفی.
- برای هر مقدار اندازه ورودی، یک درخت تصمیم داریم.
- هر مقایسه الگوریتم را به دو زیر الگوریتم تقسیم می کند.
- درخت تصمیم ترتیب مقایسه ها را بر اساس تمام ورودی های مختلف ممکن نشان می دهد.

کران پایین برای مرتب سازی مبتنی بر مقایسه

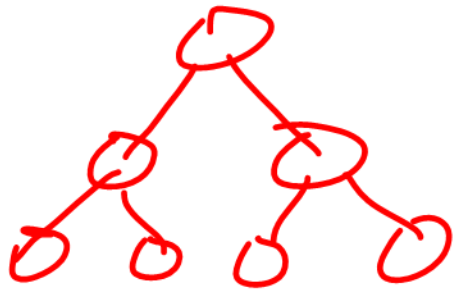
- کران پایینی برای تعداد برگ ها؟
- کران پایینی برای عمق درخت؟
- کران پایینی برای حداکثر تعداد مقایسه ها در یک اجرا؟





h	حداکثر تعداد برگ
1	1
2	2
3	4
⋮	⋮
h	2^{h-1}

عمق درخت h
تعداد برگ L



$$L \leq 2^{h-1}$$

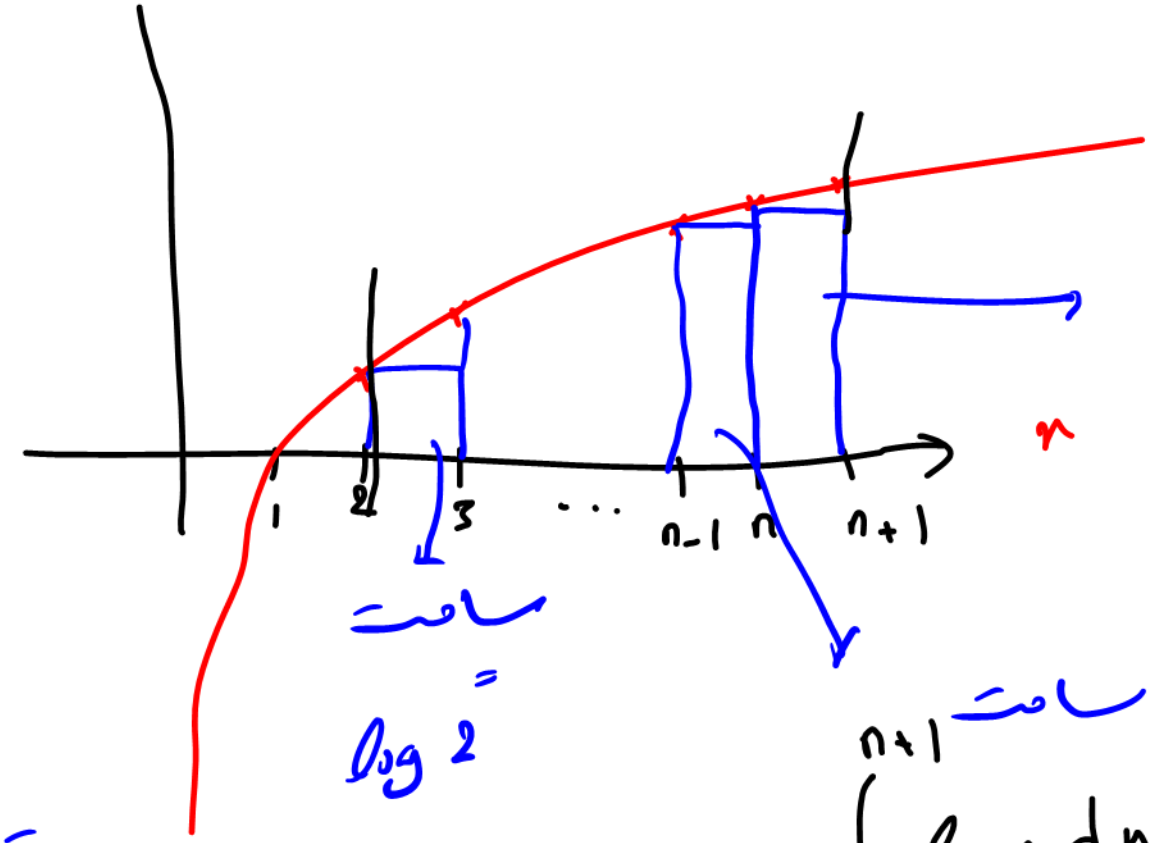
$$\log_2 L + 1 \leq h$$

$\log_2 L$ کران پایین برای عمق درخت است. <=

کران پایین برای مرتب سازی مبتنی بر مقایسه

- بیشترین تعداد مقایسه در یک اجرا برابر عمق درخت تصمیم است.
- از آنجا که n داده متمایز دارای $n!$ جایگشت هستند، برای مرتب کردن داده ها نیز به $n!$ روش نیاز است.
  - بنابراین تعداد برگ ها حداقل باید $n!$ باشد.
- می دانیم که یک درخت دودویی با عمق h حداکثر می تواند 2^h برگ داشته باشد.
  - عمق درخت حداقل باید $\log_2(n!)$ باشد.
- بنابراین $\log_2(n!)$ یک کران پایین برای تعداد مقایسه ها در الگوریتم های مرتب سازی مبتنی بر مقایسه است.

$$\log(n!) = \sum_{i=1}^{n} \log i = \sum_{i=2}^{n} \log i = A$$



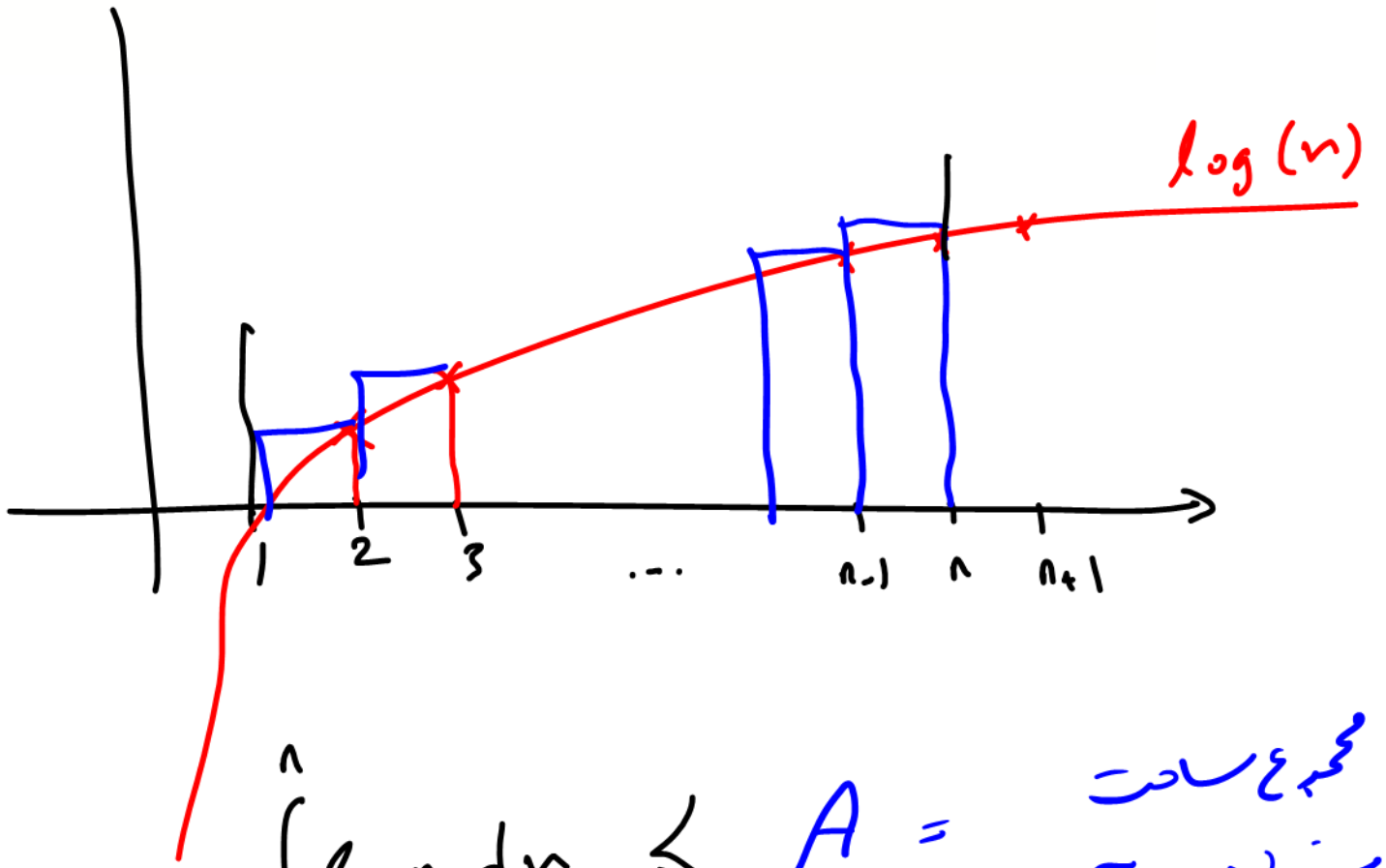
مساحت = $\log n$

مساحت = $\log 2$

مساحت = $\log(n-1)$

مجموع مساحت مستطیل ها آبی $= A \leq \int_{2}^{n+1} \ln n \, dn$

$$\log(n!) = \sum_{i=1}^{n} \log i = \sum_{i=2}^{n} \log i = A$$



$$\int_{1}^{n} \ln n \, dn \leq A =$$

مجموع ساحت مستطیل‌ها

$$\int_1^n \ln x \, dx \leq \sum_{i=2}^{n} \log i \leq \int_2^{n+1} \ln x \, dx$$

$$\int \ln x \, dx = x \ln x - x$$

$$\underbrace{n \ln n - n - (-1)}_{\Theta(n \log n)} \leq \sum_{i=2}^{n} \log i \leq \underbrace{(n+1)\log(n+1) - (n+1) - (2 \log 2 - 2)}_{\Theta(n \log n)}$$

کران پایین برای مرتب سازی مبتنی بر مقایسه

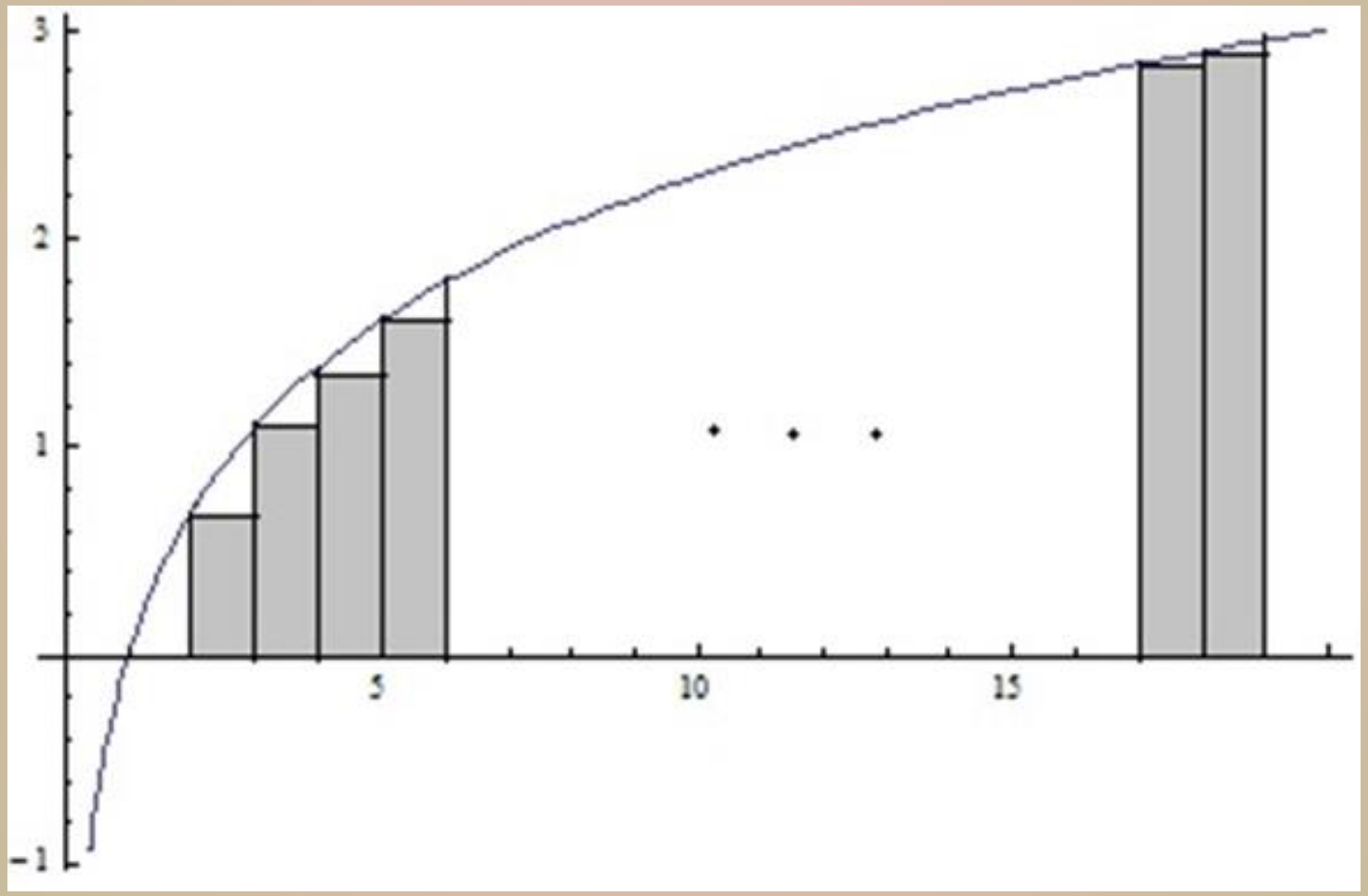
- از آنجا که تغییر مبنای لگاریتم تنها در حد یک ضریب ثابت تاثیر دارد، مبنای لگاریتم را از ۲ به عدد نپر تغییر می دهیم.

$$\ln(n!) = \ln 1 + \ln 2 + \ldots + \ln n = \sum_{i=2}^{n} \ln i$$

$$\int_{1}^{n} \ln x \, dx \leq \sum_{i=2}^{n} \ln i \leq \int_{2}^{n+1} \ln x \, dx$$

$$\Rightarrow \ln(n!) = \Theta(n \log n)$$

رابطه انتگرال با سری



محاسبه کران پایین با استفاده از تقریب استرلینگ

- تقریب استرلینگ بیان می کند که

$$\sqrt{2\pi}n^{n+\frac{1}{2}}e^{-n} \leq n! \leq en^{n+\frac{1}{2}}e^{-n}$$

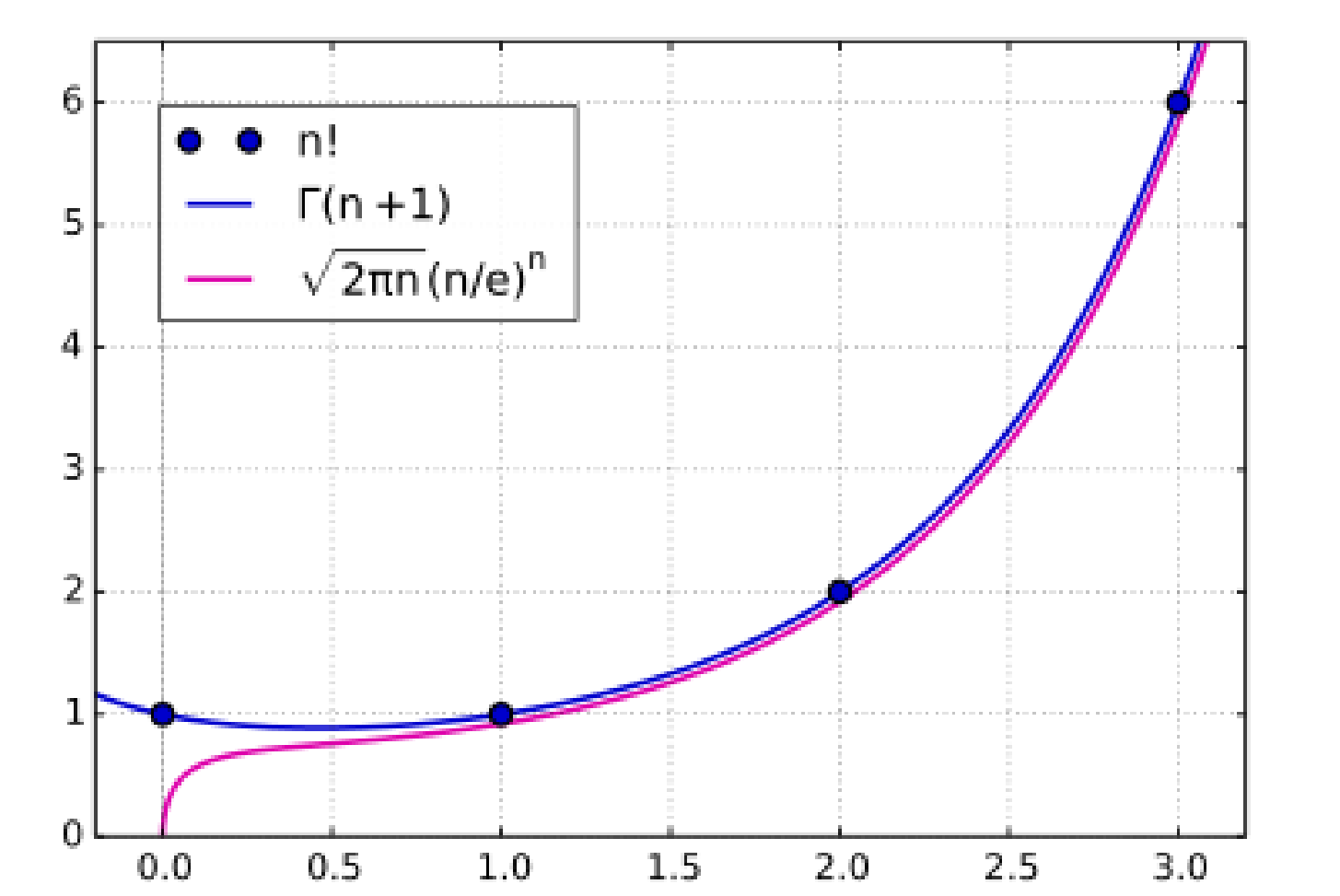
- بنابراین

$$n! \in \Theta\left(n^{n+\frac{1}{2}}e^{-n}\right)$$

- که از آن نتیجه می شود (با احتیاط خاص log)

$$\log n! \in \Theta\left(\left(n+\frac{1}{2}\right)\log n - n\right) = \Theta(n \log n)$$

تقریب استرلینگ



در مورد الگوریتم های تصادفی چه می توان گفت

- تعریف الگوریتم های تصادفی
  - الگوریتم های تصادفی در حین اجرا مقادیری تصادفی تولید می کنند و عملکرد خود را بر اساس مقادیر تصادفی تولید شده تنظیم می کنند.
- تاثیر تصادفی بودن بر روی تحلیل انجام شده
  - اگر فرض کنیم تمام مقادیر تصادفی که در حین اجرا تولید می شوند در ابتدا مشخص شده است به یک الگوریتم قطعی می رسیم که کران پایین داده شده برای آن صحیح است.
  - از آنجا که زمان اجرای الگوریتم های تصادفی برابر متوسط زمان اجرای این الگوریتم های قطعی است، کران پایین به دست آمده برای الگوریتم های تصادفی نیز صحیح است.

تمرین

- ص ۱۹۳ و ۱۹۴ کتاب CLRS
- تمرین 8.1-1
- تمرین 8.1-4

مطالعه بیشتر

- بخش ۸.۱ کتاب CLRS که از ص ۱۹۱ شروع می شود.