$T(n) \in \Theta(n)$

Quick Sort تعریف الگوریتم مرتب سازی است که در بدترین حالت $O(n^2)$ زمان نیاز دارد، اما در حالت میانگین $O(n \log n)$ زمان نیاز دارد. این الگوریتم یکی از سریع‌ترین الگوریتم‌های مرتب سازی است که در عمل استفاده می‌شود.

Selection Sort

الگوریتم مرتب سازی است که در هر مرحله کوچکترین عنصر را پیدا می‌کند و آن را در جای خود قرار می‌دهد. این الگوریتم در بدترین حالت $O(n^2)$ زمان نیاز دارد.

$$T(n) = T(\frac{n}{5}) + T(\frac{7n}{10}) + \Theta(n)$$

$$T(n) \in \Theta(n)$$

مرتب سازی زمان خطی

Sorting in linear time

* الگوریتم هایی مرتب سازی که با حال بدترین حالت
بدترین حالت $\Omega(n \lg n)$ بیان داشتند

مرتب سورت $\theta(n \lg n)$

مرتب سازی سریع $\Omega(n \lg n) - O(n^2)$

مرتب سازی درجی $\Omega(n) - O(n^2)$ بهترین حالت

* بعضی الگوریتم‌ها یی که تا حال دیدیم

در بهترین حالت زمان حداقل $n \log n$

داشته اند.

قضیه:

" الگوریتم‌های مرتب سازی که براساس مقایسه

کار می‌کنند، در بهترین حالت دارای

زمان $\Omega(n \log n)$ هستند "

یک اثبات کلی و مشهور برای این قضیه :

n عدد مختلف دارای $n!$ ترتیب مختلف است

که فقط یکی از این $n!$ ترتیب ، ترتیب

مرتب شده صعودی است که صفواهیم آن را

پیدا کنیم . با مقایسه $\underline{2}$ عدد جایگاه

و ترتیب این دو عدد در هر ترتیب n عدد

مشخص میشود : $a<b$: ..... a .... b ---

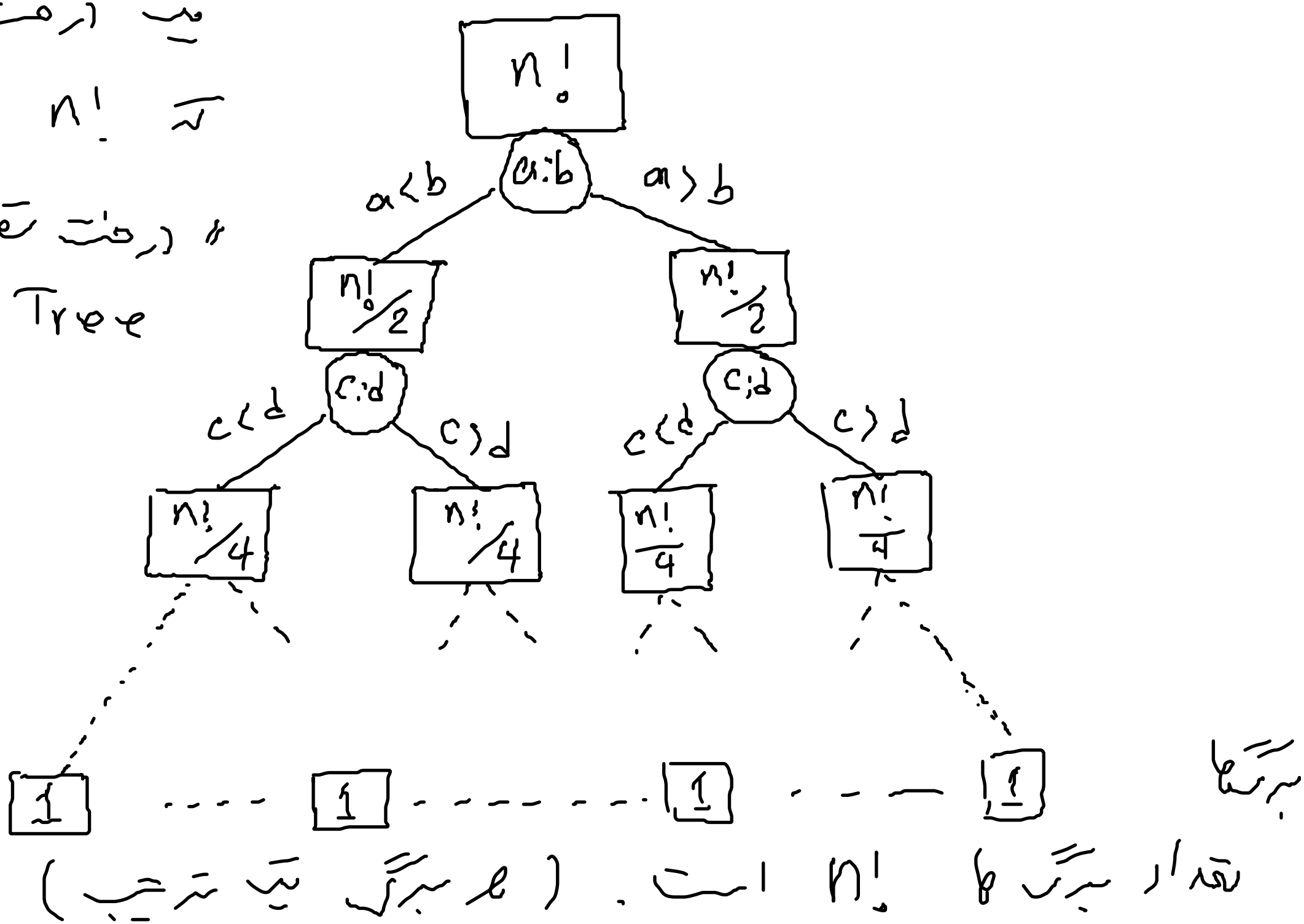
$a>b$: ..... b ... a ...

با یک مقایسه بین دو عدد نصف از ترتیب ها معتبر و نیمه دیگر نامعتبر است و باید کنار گذاشته شود.

یک درخت باینری که n! برگ دارد.

" درخت تصمیم "

Decision Tree



برگ‌ها

تعداد برگ‌ها n! است. (هر برگ یک ترتیب)

الگوریتم مرتب سازی برای اینکه به جواب برسد
از درخت تصمیم یک مسیر از ریشه به برگ
طی میکند. در بدترین حالت این مسیر طولانی ترین
مسیر است. بنابراین

"زمان الگوریتم در بدترین حالت
متناسب با ارتفاع درخت تصمیم است."

کمترین ارتفاع درخت باینری وقتی است که بالانس
باشد. پس کمترین ارتفاع درخت تصمیم وقتی
که بالانس است حاوی (n!) برگ است.

الگوریتم هایی برای مرتب سازی ارائه میکنیم که
مستقل بر مقایسه نیستند و زمان کمتر از $n \lg n$
میتوانند داشته باشند.

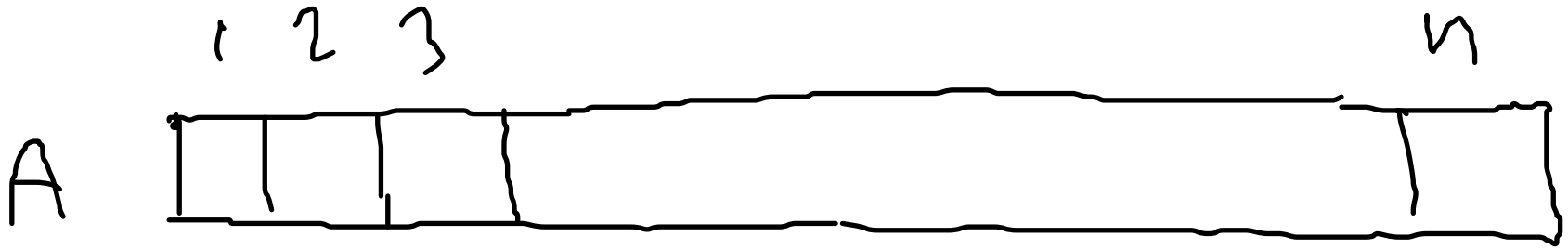
الگوریتم Sort (counting) بشمارشی

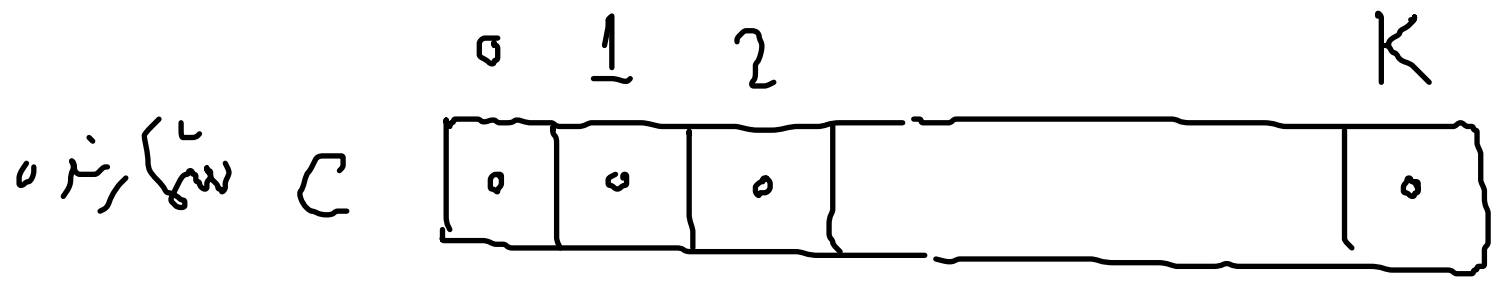
Radix sort "

Bucket sort "

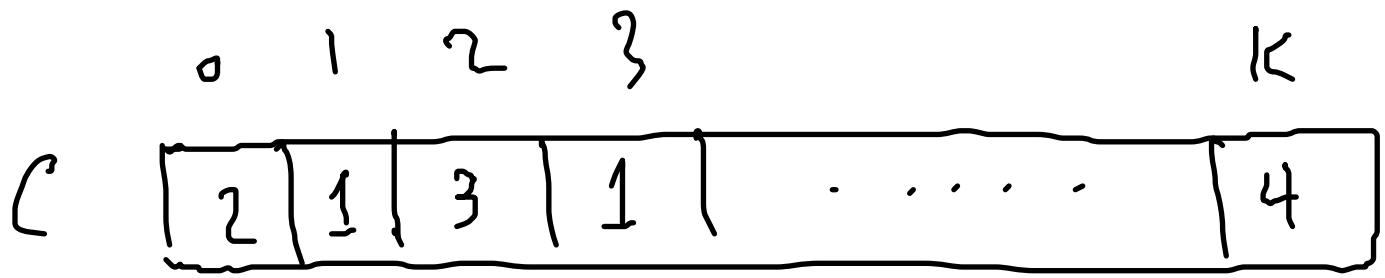
الگوریتم Counting sort :

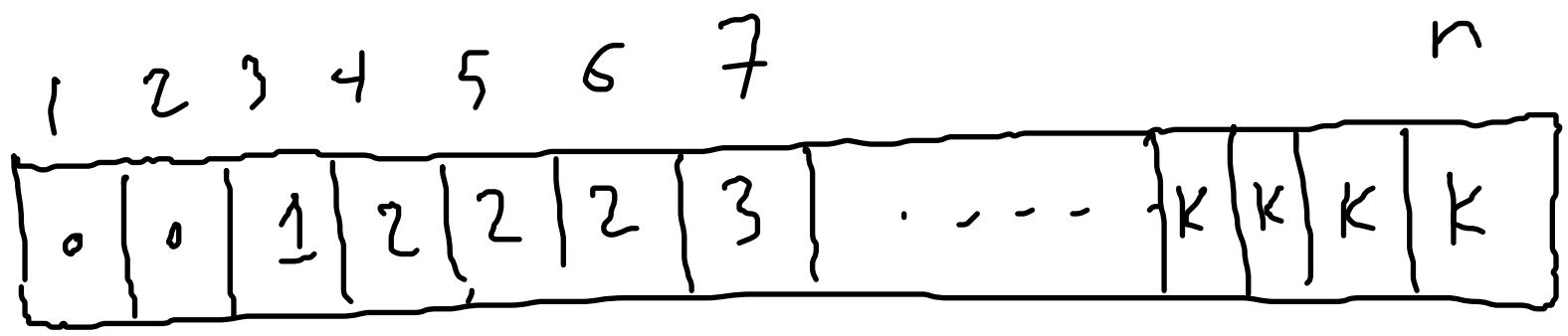
برای مرتب سازی اعداد صحیح در بازه
۰ تا K استفاده میشود.





C [A [i]] + +

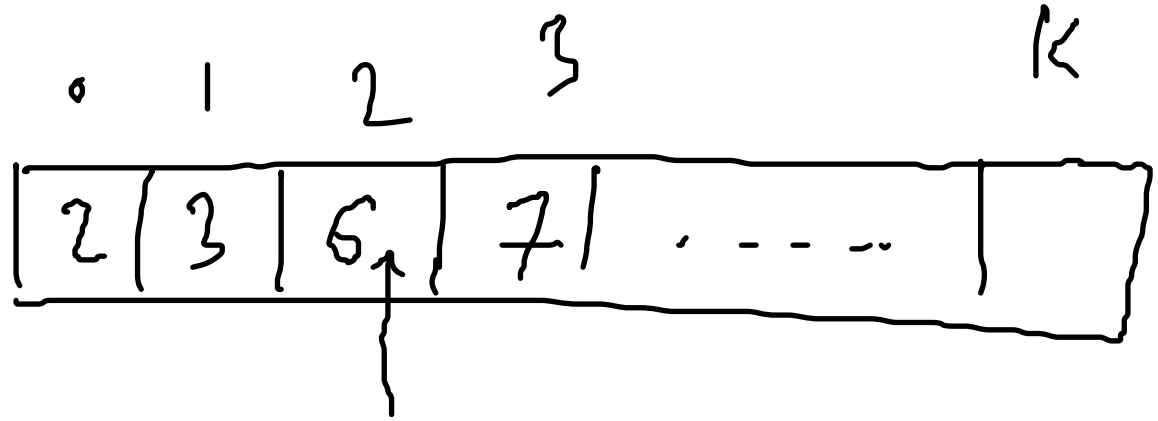




آرایه صورت‌گرفته

این

نهایی



حال تکه‌ای از

آرایه C حاصل

می‌کنیم

یعنی آخرین عدد 2 در آرایه

خروجی در خانه 6 ام آرایه

قرار می‌گیرد

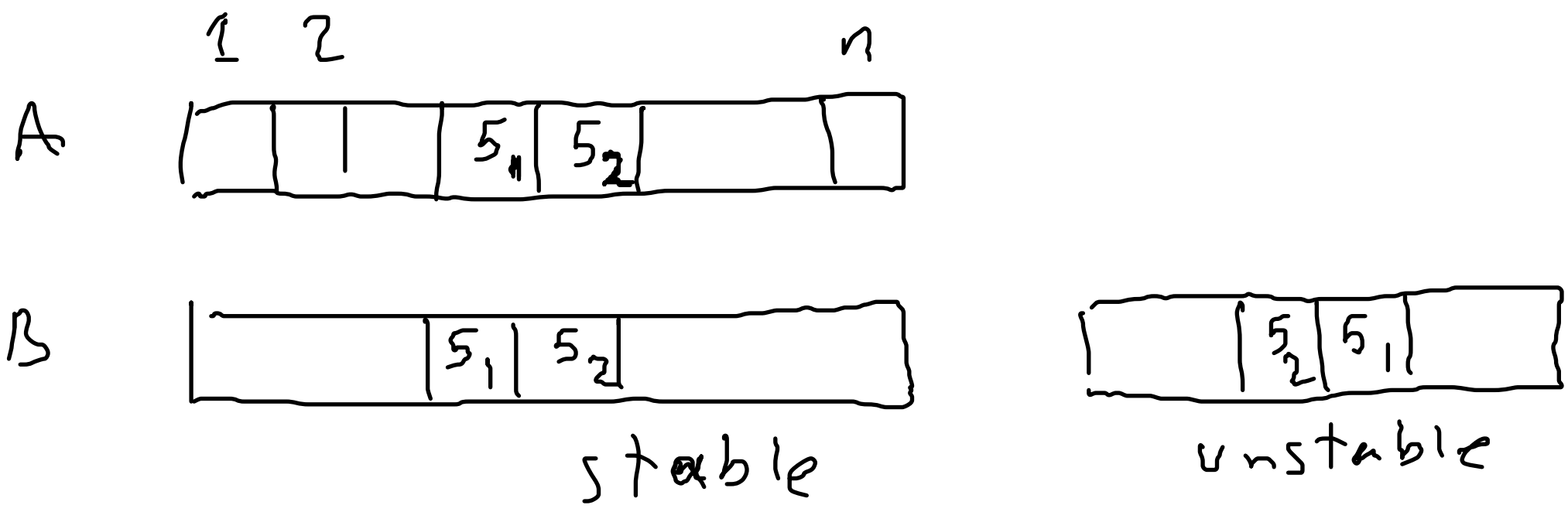
```
Counting_sort(A, n, k)
1. for i ← 0 to k do                    c1 (k+1)
2.     C[i] ← 0
3. for i ← 1 to n do                    c2 n
4.     C[A[i]]++
5. for i ← 1 to k do                    c3 k
6.     C[i] = C[i-1] + C[i]
7. for i ← n down to 1 do
8.     B[C[A[i]]] ← A[i]                c4 n
9.     C[A[i]]--
```

التعقيد الزمني = $c_1(k+1) + c_2 n + c_3 k + c_4 n$

$\in \theta(n+k)$

مرتب سازی پایدار Stable

یک الگوریتم مرتب سازی پایدار یا stable است اگر ترتیب اعداد مساوی هم در ورودی حفظ شود یا نشود



مثال) A حدد عناصر المصفوفة B التي تساوي 3 - 0

	1	2	3	4	5	6	7	8
A	3	0	1	2	3	0	1	0

C

0	1	2	3
~~0~~	~~0~~	~~0~~	~~0~~
1	1	1	1
2	2	2	2
3			

$\Rightarrow$

C

0	1	2	3
3	2	1	2

C

0	1	2	3
~~3~~	~~5~~	~~6~~	~~8~~
2	4	3	5	7	1	8
					0	

B

1	2	3	4	5	6	7	8
0	0	0	1	1	2	3	3