QuickSort – animace

pro studenty učitelství informatiky

Martina Šimůnková

Katedra matematiky, FP TUL

29. dubna 2024

Popis algoritmu

V souboru, který máme seřadit vybereme jeden prvek a nazveme ho *pivotem*.

5	12	7	4	17	10	3	11	9	2	8

Popis algoritmu

V této ukázce vybereme za *pivota* poslední prvek souboru.

5	12	7	4	17	10	3	11	9	2	8

Popis algoritmu

Index i nad prvním prvkem souboru posouváme, dokud nanarazíme na prvek **větší** než pivot.

$i \rightarrow$

5	12	7	4	17	10	3	11	9	2	8

Popis algoritmu

Index i nad prvním prvkem souboru posouváme, dokud nanarazíme na prvek **větší** než pivot.

	$i \rightarrow$									
5	12	7	4	17	10	3	11	9	2	8

Popis algoritmu

Index j nad posledním prvkem souboru posouváme, dokud nanarazíme na prvek **menší** než pivot.

	$i \rightarrow$								$\leftarrow j$	
5	12	7	4	17	10	3	11	9	2	8

Popis algoritmu

Prvky pod těmito indexy

	$i \rightarrow$								$\leftarrow j$	
5	12	7	4	17	10	3	11	9	2	8

Popis algoritmu

Prvky pod těmito indexy vyměníme.

	$i \rightarrow$								$\leftarrow j$	
5	2	7	4	17	10	3	11	9	12	8

Popis algoritmu

Posuneme oba indexy podle stejného pravidla jako dříve.

				$i \rightarrow$		$\leftarrow j$				
5	2	7	4	17	10	3	11	9	12	8

Popis algoritmu

Prvky vyměníme.

				$i \rightarrow$		$\leftarrow j$				
5	2	7	4	3	10	17	11	9	12	8

Popis algoritmu

Posuneme indexy.

				$\leftarrow j$	$i \rightarrow$					
5	2	7	4	3	10	17	11	9	12	8

Popis algoritmu

V okamžiku, kdy bude index i vpravo od indexu j, vyměníme prvek pod i s pivotem.

				$\leftarrow j$	$i \rightarrow$					
5	2	7	4	3	8	17	11	9	12	10

Popis algoritmu

Končí první fáze algoritmu. Všimněte si, že pivot nyní dělí soubor na menší prvky (ty jsou vlevo od něj) a větší prvky (ty jsou vpravo).

5 2 7 4 3 8 17 11 9 12 10

Popis algoritmu

Z pivota se stane rozhraní dvou souborů.

5 2 7 4 3 8 17 11 9 12 10

Popis algoritmu

Seřadíme levou část stejným algoritmem.

5 2 7 4 3 8 17 11 9 12 10

Popis algoritmu

Seřadíme levou část stejným algoritmem.

$i \rightarrow$

5	2	7	4	3	8	17	11	9	12	10

Popis algoritmu

Seřadíme levou část stejným algoritmem.

$i \rightarrow$ $\leftarrow j$

5	2	7	4	3	8	17	11	9	12	10

Popis algoritmu

Seřadíme levou část stejným algoritmem.

$i \rightarrow$				$\leftarrow j$						
5	2	7	4	3	8	17	11	9	12	10

Popis algoritmu

Seřadíme levou část stejným algoritmem.

$i \rightarrow$				$\leftarrow j$					
2	5	7	4	3	8	17	11	9	12	10

Popis algoritmu

Seřadíme levou část stejným algoritmem.

$\leftarrow j \quad i \rightarrow$

2	5	7	4	3	8	17	11	9	12	10

Popis algoritmu

Seřadíme levou část stejným algoritmem.

$\leftarrow j \quad i \rightarrow$

2	3	7	4	5	8	17	11	9	12	10

Popis algoritmu

Pivot je opět rozhraní menších souborů.

2 3 7 4 5 8 17 11 9 12 10

Popis algoritmu

První část je seřazená, druhou seřadíme quick sortem, případně i pomalejším algoritmem.

2 3 7 4 5 8 17 11 9 12 10

Popis algoritmu

První část je seřazená, druhou seřadíme quick sortem, případně i pomalejším algoritmem.

2 3 4 5 7 8 17 11 9 12 10

Popis algoritmu

Ještě quick sortem seřadíme pravou část souboru.

2 3 4 5 7 8 17 11 9 12 10

Popis algoritmu

Ještě quick sortem seřadíme pravou část souboru.

$i \rightarrow$

2	3	4	5	7	8	17	11	9	12	10

Popis algoritmu

Ještě quick sortem seřadíme pravou část souboru.

						$i \rightarrow$			$\leftarrow j$	
2	3	4	5	7	8	17	11	9	12	10

Popis algoritmu

Ještě quick sortem seřadíme pravou část souboru.

						$i \rightarrow$		$\leftarrow j$		
2	3	4	5	7	8	17	11	9	12	10

Popis algoritmu

Ještě quick sortem seřadíme pravou část souboru.

						$i \rightarrow$		$\leftarrow j$		
2	3	4	5	7	8	9	11	17	12	10

Popis algoritmu

Ještě quick sortem seřadíme pravou část souboru.

						$\leftarrow j$	$i \rightarrow$			
2	3	4	5	7	8	9	11	17	12	10

Popis algoritmu

Ještě quick sortem seřadíme pravou část souboru.

						$\leftarrow j$	$i \rightarrow$			
2	3	4	5	7	8	9	10	17	12	11

Popis algoritmu

Ještě quick sortem seřadíme pravou část souboru.

2 3 4 5 7 8 9 10 17 12 11

Popis algoritmu

Malé soubory seřadíme jakýmkoliv algoritmem (například insertsortem).

2 3 4 5 7 8 9 10 17 12 11

Popis algoritmu

Malé soubory seřadíme jakýmkoliv algoritmem (například insertsortem).

2 3 4 5 7 8 9 10 11 12 17

Výklad algoritmu jsem převzala od studenta Ondřeje Svárovského, který ho předvedl při zkoušení.

Na předchozím slajdu jsme ukázali, jak quick sort pracuje. Zároveň jsme předvedli, že při běhu algoritmu není třeba soubor kopírovat – dá se provést „na místě“. (Na rozdíl od merge sortu.)

Časová složitost Quick sortu

Nyní se budeme věnovat rozboru časové složitosti quicksortu.

Projdeme první část algoritmu. Všimněte si, že tato část má počet operací úměrný velikosti vstupu. Má tedy časovou složitost $O(n)$.

Projdeme druhou část algoritmu. Všimněte si, že i tato část má počet operací úměrný velikosti vstupu, a tedy časovou složitost $O(n)$.

Kolik bude takových částí? Pokud by pivot dělil soubor na zhruba stejně velké části, bude, podobně jako u merge sortu, částí řádově $\log_2(n)$. Celková časová složitost je tedy $O(n \log(n))$.

Poslední částí je dotřídění krátkých úseků.

Časová složitost Quick sortu

Nyní se budeme věnovat rozboru časové složitosti quicksortu. Projdeme první část algoritmu. Všimněte si, že tato část má počet operací úměrný velikosti vstupu. Má tedy časovou složitost $O(n)$.

5 12 7 4 17 10 3 11 9 2 8

Projdeme druhou část algoritmu. Všimněte si, že i tato část má počet operací úměrný velikosti vstupu, a tedy časovou složitost $O(n)$.

Kolik bude takových částí? Pokud by pivot dělil soubor na zhruba stejně velké části, bude, podobně jako u merge sortu, částí řádově $\log_2(n)$. Celková časová složitost je tedy $O(n \log(n))$.

Poslední částí je dotřídění krátkých úseků.

Časová složitost Quick sortu

Nyní se budeme věnovat rozboru časové složitosti quicksortu.

Projdeme první část algoritmu. Všimněte si, že tato část má počet operací úměrný velikosti vstupu. Má tedy časovou složitost $O(n)$.

$i \rightarrow$

5	12	7	4	17	10	3	11	9	2	8

Projdeme druhou část algoritmu. Všimněte si, že i tato část má počet operací úměrný velikosti vstupu, a tedy časovou složitost $O(n)$.

Kolik bude takových částí? Pokud by pivot dělil soubor na zhruba stejně velké části, bude, podobně jako u merge sortu, částí řádově $\log_2(n)$. Celková časová složitost je tedy $O(n \log(n))$.

Poslední částí je dotřídění krátkých úseků.

Časová složitost Quick sortu

Nyní se budeme věnovat rozboru časové složitosti quicksortu.

Projdeme první část algoritmu. Všimněte si, že tato část má počet operací úměrný velikosti vstupu. Má tedy časovou složitost $O(n)$.

Projdeme druhou část algoritmu. Všimněte si, že i tato část má počet operací úměrný velikosti vstupu, a tedy časovou složitost $O(n)$.

Kolik bude takových částí? Pokud by pivot dělil soubor na zhruba stejně velké části, bude, podobně jako u merge sortu, částí řádově $\log_2(n)$. Celková časová složitost je tedy $O(n \log(n))$.

Poslední částí je dotřídění krátkých úseků.

Časová složitost Quick sortu

Nyní se budeme věnovat rozboru časové složitosti quicksortu.

Projdeme první část algoritmu. Všimněte si, že tato část má počet operací úměrný velikosti vstupu. Má tedy časovou složitost $O(n)$.

Projdeme druhou část algoritmu. Všimněte si, že i tato část má počet operací úměrný velikosti vstupu, a tedy časovou složitost $O(n)$.

Kolik bude takových částí? Pokud by pivot dělil soubor na zhruba stejně velké části, bude, podobně jako u merge sortu, částí řádově $\log_2(n)$. Celková časová složitost je tedy $O(n \log(n))$.

Poslední částí je dotřídění krátkých úseků.

Časová složitost Quick sortu

Nyní se budeme věnovat rozboru časové složitosti quicksortu.

Projdeme první část algoritmu. Všimněte si, že tato část má počet operací úměrný velikosti vstupu. Má tedy časovou složitost $O(n)$.

Projdeme druhou část algoritmu. Všimněte si, že i tato část má počet operací úměrný velikosti vstupu, a tedy časovou složitost $O(n)$.

Kolik bude takových částí? Pokud by pivot dělil soubor na zhruba stejně velké části, bude, podobně jako u merge sortu, částí řádově $\log_2(n)$. Celková časová složitost je tedy $O(n \log(n))$.

Poslední částí je dotřídění krátkých úseků.

Časová složitost Quick sortu

Nyní se budeme věnovat rozboru časové složitosti quicksortu.

Projdeme první část algoritmu. Všimněte si, že tato část má počet operací úměrný velikosti vstupu. Má tedy časovou složitost $O(n)$.

Projdeme druhou část algoritmu. Všimněte si, že i tato část má počet operací úměrný velikosti vstupu, a tedy časovou složitost $O(n)$.

Kolik bude takových částí? Pokud by pivot dělil soubor na zhruba stejně velké části, bude, podobně jako u merge sortu, částí řádově $\log_2(n)$. Celková časová složitost je tedy $O(n \log(n))$.

Poslední částí je dotřídění krátkých úseků.

Časová složitost Quick sortu

Nyní se budeme věnovat rozboru časové složitosti quicksortu.

Projdeme první část algoritmu. Všimněte si, že tato část má počet operací úměrný velikosti vstupu. Má tedy časovou složitost $O(n)$.

Projdeme druhou část algoritmu. Všimněte si, že i tato část má počet operací úměrný velikosti vstupu, a tedy časovou složitost $O(n)$.

Kolik bude takových částí? Pokud by pivot dělil soubor na zhruba stejně velké části, bude, podobně jako u merge sortu, částí řádově $\log_2(n)$. Celková časová složitost je tedy $O(n \log(n))$.

Poslední částí je dotřídění krátkých úseků.

Časová složitost Quick sortu

Nyní se budeme věnovat rozboru časové složitosti quicksortu.

Projdeme první část algoritmu. Všimněte si, že tato část má počet operací úměrný velikosti vstupu. Má tedy časovou složitost $O(n)$.

Projdeme druhou část algoritmu. Všimněte si, že i tato část má počet operací úměrný velikosti vstupu, a tedy časovou složitost $O(n)$.

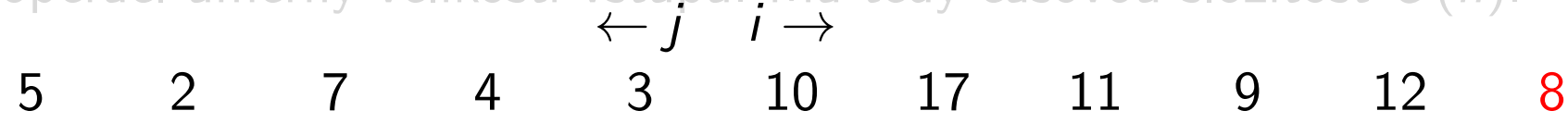
Kolik bude takových částí? Pokud by pivot dělil soubor na zhruba stejně velké části, bude, podobně jako u merge sortu, částí řádově $\log_2(n)$. Celková časová složitost je tedy $O(n \log(n))$.

Poslední částí je dotřídění krátkých úseků.

Časová složitost Quick sortu

Nyní se budeme věnovat rozboru časové složitosti quicksortu.

Projdeme první část algoritmu. Všimněte si, že tato část má počet operací úměrný velikosti vstupu. Má tedy časovou složitost $O(n)$.



Projdeme druhou část algoritmu. Všimněte si, že i tato část má počet operací úměrný velikosti vstupu, a tedy časovou složitost $O(n)$.

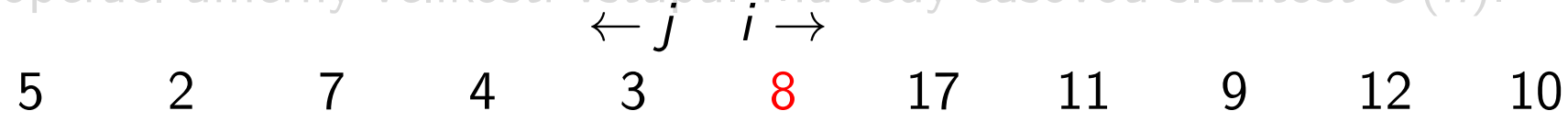
Kolik bude takových částí? Pokud by pivot dělil soubor na zhruba stejně velké části, bude, podobně jako u merge sortu, částí řádově $\log_2(n)$. Celková časová složitost je tedy $O(n \log(n))$.

Poslední částí je dotřídění krátkých úseků.

Časová složitost Quick sortu

Nyní se budeme věnovat rozboru časové složitosti quicksortu.

Projdeme první část algoritmu. Všimněte si, že tato část má počet operací úměrný velikosti vstupu. Má tedy časovou složitost $O(n)$.



Projdeme druhou část algoritmu. Všimněte si, že i tato část má počet operací úměrný velikosti vstupu, a tedy časovou složitost $O(n)$.

Kolik bude takových částí? Pokud by pivot dělil soubor na zhruba stejně velké části, bude, podobně jako u merge sortu, částí řádově $\log_2(n)$. Celková časová složitost je tedy $O(n \log(n))$.

Poslední částí je dotřídění krátkých úseků.

Časová složitost Quick sortu

Nyní se budeme věnovat rozboru časové složitosti quicksortu.
Projdeme první část algoritmu. Všimněte si, že tato část má počet operací úměrný velikosti vstupu. Má tedy časovou složitost $O(n)$.

5 2 7 4 3 8 17 11 9 12 10

Projdeme druhou část algoritmu. Všimněte si, že i tato část má počet operací úměrný velikosti vstupu, a tedy časovou složitost $O(n)$.

Kolik bude takových částí? Pokud by pivot dělil soubor na zhruba stejně velké části, bude, podobně jako u merge sortu, částí řádově $\log_2(n)$. Celková časová složitost je tedy $O(n \log(n))$.

Poslední částí je dotřídění krátkých úseků.

Časová složitost Quick sortu

Nyní se budeme věnovat rozboru časové složitosti quicksortu. Projdeme první část algoritmu. Všimněte si, že tato část má počet operací úměrný velikosti vstupu. Má tedy časovou složitost $O(n)$.

5 2 7 4 3 8 17 11 9 12 10

Projdeme druhou část algoritmu. Všimněte si, že i tato část má počet operací úměrný velikosti vstupu, a tedy časovou složitost $O(n)$.

Kolik bude takových částí? Pokud by pivot dělil soubor na zhruba stejně velké části, bude, podobně jako u merge sortu, částí řádově $\log_2(n)$. Celková časová složitost je tedy $O(n \log(n))$.

Poslední částí je dotřídění krátkých úseků.

Časová složitost Quick sortu

Nyní se budeme věnovat rozboru časové složitosti quicksortu. Projdeme první část algoritmu. Všimněte si, že tato část má počet operací úměrný velikosti vstupu. Má tedy časovou složitost $O(n)$.

5 2 7 4 3 8 17 11 9 12 10

Projdeme druhou část algoritmu. Všimněte si, že i tato část má počet operací úměrný velikosti vstupu, a tedy časovou složitost $O(n)$.

5 2 7 4 3 8 17 11 9 12 10

Kolik bude takových částí? Pokud by pivot dělil soubor na zhruba stejně velké části, bude, podobně jako u merge sortu, částí řádově $\log_2(n)$. Celková časová složitost je tedy $O(n \log(n))$.

Poslední částí je dotřídění krátkých úseků.

Časová složitost Quick sortu

Nyní se budeme věnovat rozboru časové složitosti quicksortu. Projdeme první část algoritmu. Všimněte si, že tato část má počet operací úměrný velikosti vstupu. Má tedy časovou složitost $O(n)$.

5	2	7	4	3	8	17	11	9	12	10

Projdeme druhou část algoritmu. Všimněte si, že i tato část má počet operací úměrný velikosti vstupu, a tedy časovou složitost $O(n)$.

$i \rightarrow$

5	2	7	4	3	8	17	11	9	12	10

Kolik bude takových částí? Pokud by pivot dělil soubor na zhruba stejně velké části, bude, podobně jako u merge sortu, částí řádově $\log_2(n)$. Celková časová složitost je tedy $O(n \log(n))$.

Poslední částí je dotřídění krátkých úseků.

Časová složitost Quick sortu

Nyní se budeme věnovat rozboru časové složitosti quicksortu. Projdeme první část algoritmu. Všimněte si, že tato část má počet operací úměrný velikosti vstupu. Má tedy časovou složitost $O(n)$.

5	2	7	4	3	8	17	11	9	12	10

Projdeme druhou část algoritmu. Všimněte si, že i tato část má počet operací úměrný velikosti vstupu, a tedy časovou složitost $O(n)$.

$i \rightarrow$			$\leftarrow j$							
5	2	7	4	3	8	17	11	9	12	10

Kolik bude takových částí? Pokud by pivot dělil soubor na zhruba stejně velké části, bude, podobně jako u merge sortu, částí řádově $\log_2(n)$. Celková časová složitost je tedy $O(n \log(n))$.

Poslední částí je dotřídění krátkých úseků.

Časová složitost Quick sortu

Nyní se budeme věnovat rozboru časové složitosti quicksortu. Projdeme první část algoritmu. Všimněte si, že tato část má počet operací úměrný velikosti vstupu. Má tedy časovou složitost $O(n)$.

5 2 7 4 3 8 17 11 9 12 10

Projdeme druhou část algoritmu. Všimněte si, že i tato část má počet operací úměrný velikosti vstupu, a tedy časovou složitost $O(n)$.

$i \rightarrow \quad \leftarrow j$

5 2 7 4 3 8 17 11 9 12 10

Kolik bude takových částí? Pokud by pivot dělil soubor na zhruba stejně velké části, bude, podobně jako u merge sortu, částí řádově $\log_2(n)$. Celková časová složitost je tedy $O(n \log(n))$.

Poslední částí je dotřídění krátkých úseků.

Časová složitost Quick sortu

Nyní se budeme věnovat rozboru časové složitosti quicksortu. Projdeme první část algoritmu. Všimněte si, že tato část má počet operací úměrný velikosti vstupu. Má tedy časovou složitost $O(n)$.

5 2 7 4 3 8 17 11 9 12 10

Projdeme druhou část algoritmu. Všimněte si, že i tato část má počet operací úměrný velikosti vstupu, a tedy časovou složitost $O(n)$.

$i \rightarrow \quad \leftarrow j$

2 5 7 4 3 8 17 11 9 12 10

Kolik bude takových částí? Pokud by pivot dělil soubor na zhruba stejně velké části, bude, podobně jako u merge sortu, částí řádově $\log_2(n)$. Celková časová složitost je tedy $O(n \log(n))$.

Poslední částí je dotřídění krátkých úseků.

Časová složitost Quick sortu

Nyní se budeme věnovat rozboru časové složitosti quicksortu. Projdeme první část algoritmu. Všimněte si, že tato část má počet operací úměrný velikosti vstupu. Má tedy časovou složitost $O(n)$.

5	2	7	4	3	8	17	11	9	12	10

Projdeme druhou část algoritmu. Všimněte si, že i tato část má počet operací úměrný velikosti vstupu, a tedy časovou složitost $O(n)$.

$\leftarrow j \quad i \rightarrow$

2	5	7	4	3	8	17	11	9	12	10

Kolik bude takových částí? Pokud by pivot dělil soubor na zhruba stejně velké části, bude, podobně jako u merge sortu, částí řádově $\log_2(n)$. Celková časová složitost je tedy $O(n \log(n))$.

Poslední částí je dotřídění krátkých úseků.

Časová složitost Quick sortu

Nyní se budeme věnovat rozboru časové složitosti quicksortu. Projdeme první část algoritmu. Všimněte si, že tato část má počet operací úměrný velikosti vstupu. Má tedy časovou složitost $O(n)$.

5	2	7	4	3	8	17	11	9	12	10

Projdeme druhou část algoritmu. Všimněte si, že i tato část má počet operací úměrný velikosti vstupu, a tedy časovou složitost $O(n)$.

$\leftarrow j \quad i \rightarrow$

2	3	7	4	5	8	17	11	9	12	10

Kolik bude takových částí? Pokud by pivot dělil soubor na zhruba stejně velké části, bude, podobně jako u merge sortu, částí řádově $\log_2(n)$. Celková časová složitost je tedy $O(n \log(n))$.

Poslední částí je dotřídění krátkých úseků.

Časová složitost Quick sortu

Nyní se budeme věnovat rozboru časové složitosti quicksortu. Projdeme první část algoritmu. Všimněte si, že tato část má počet operací úměrný velikosti vstupu. Má tedy časovou složitost $O(n)$.

5 2 7 4 3 8 17 11 9 12 10

Projdeme druhou část algoritmu. Všimněte si, že i tato část má počet operací úměrný velikosti vstupu, a tedy časovou složitost $O(n)$.

2 3 7 4 5 8 17 11 9 12 10

Kolik bude takových částí? Pokud by pivot dělil soubor na zhruba stejně velké části, bude, podobně jako u merge sortu, částí řádově $\log_2(n)$. Celková časová složitost je tedy $O(n \log(n))$.

Poslední částí je dotřídění krátkých úseků.

Časová složitost Quick sortu

Nyní se budeme věnovat rozboru časové složitosti quicksortu. Projdeme první část algoritmu. Všimněte si, že tato část má počet operací úměrný velikosti vstupu. Má tedy časovou složitost $O(n)$.

5 2 7 4 3 8 17 11 9 12 10

Projdeme druhou část algoritmu. Všimněte si, že i tato část má počet operací úměrný velikosti vstupu, a tedy časovou složitost $O(n)$.

2 3 7 4 5 8 17 11 9 12 10

Kolik bude takových částí? Pokud by pivot dělil soubor na zhruba stejně velké části, bude, podobně jako u merge sortu, částí řádově $\log_2(n)$. Celková časová složitost je tedy $O(n \log(n))$.

Poslední částí je dotřídění krátkých úseků.

Časová složitost Quick sortu

Nyní se budeme věnovat rozboru časové složitosti quicksortu. Projdeme první část algoritmu. Všimněte si, že tato část má počet operací úměrný velikosti vstupu. Má tedy časovou složitost $O(n)$.

5	2	7	4	3	8	17	11	9	12	10

Projdeme druhou část algoritmu. Všimněte si, že i tato část má počet operací úměrný velikosti vstupu, a tedy časovou složitost $O(n)$.

						$i \rightarrow$				
2	3	7	4	5	8	17	11	9	12	10

Kolik bude takových částí? Pokud by pivot dělil soubor na zhruba stejně velké části, bude, podobně jako u merge sortu, částí řádově $\log_2(n)$. Celková časová složitost je tedy $O(n \log(n))$.

Poslední částí je dotřídění krátkých úseků.

Časová složitost Quick sortu

Nyní se budeme věnovat rozboru časové složitosti quicksortu. Projdeme první část algoritmu. Všimněte si, že tato část má počet operací úměrný velikosti vstupu. Má tedy časovou složitost $O(n)$.

5	2	7	4	3	8	17	11	9	12	10

Projdeme druhou část algoritmu. Všimněte si, že i tato část má počet operací úměrný velikosti vstupu, a tedy časovou složitost $O(n)$.

						$i \rightarrow$			$\leftarrow j$	
2	3	7	4	5	8	17	11	9	12	10

Kolik bude takových částí? Pokud by pivot dělil soubor na zhruba stejně velké části, bude, podobně jako u merge sortu, částí řádově $\log_2(n)$. Celková časová složitost je tedy $O(n \log(n))$.

Poslední částí je dotřídění krátkých úseků.

Časová složitost Quick sortu

Nyní se budeme věnovat rozboru časové složitosti quicksortu. Projdeme první část algoritmu. Všimněte si, že tato část má počet operací úměrný velikosti vstupu. Má tedy časovou složitost $O(n)$.

5	2	7	4	3	8	17	11	9	12	10

Projdeme druhou část algoritmu. Všimněte si, že i tato část má počet operací úměrný velikosti vstupu, a tedy časovou složitost $O(n)$.

						$i \rightarrow$		$\leftarrow j$		
2	3	7	4	5	8	17	11	9	12	10

Kolik bude takových částí? Pokud by pivot dělil soubor na zhruba stejně velké části, bude, podobně jako u merge sortu, částí řádově $\log_2(n)$. Celková časová složitost je tedy $O(n \log(n))$.

Poslední částí je dotřídění krátkých úseků.

Časová složitost Quick sortu

Nyní se budeme věnovat rozboru časové složitosti quicksortu. Projdeme první část algoritmu. Všimněte si, že tato část má počet operací úměrný velikosti vstupu. Má tedy časovou složitost $O(n)$.

5	2	7	4	3	8	17	11	9	12	10

Projdeme druhou část algoritmu. Všimněte si, že i tato část má počet operací úměrný velikosti vstupu, a tedy časovou složitost $O(n)$.

						$i \rightarrow$		$\leftarrow j$		
2	3	7	4	5	8	9	11	17	12	10

Kolik bude takových částí? Pokud by pivot dělil soubor na zhruba stejně velké části, bude, podobně jako u merge sortu, částí řádově $\log_2(n)$. Celková časová složitost je tedy $O(n \log(n))$.

Poslední částí je dotřídění krátkých úseků.

Časová složitost Quick sortu

Nyní se budeme věnovat rozboru časové složitosti quicksortu. Projdeme první část algoritmu. Všimněte si, že tato část má počet operací úměrný velikosti vstupu. Má tedy časovou složitost $O(n)$.

5	2	7	4	3	8	17	11	9	12	10

Projdeme druhou část algoritmu. Všimněte si, že i tato část má počet operací úměrný velikosti vstupu, a tedy časovou složitost $O(n)$.

						$\leftarrow j$	$i \rightarrow$			
2	3	7	4	5	8	9	11	17	12	10

Kolik bude takových částí? Pokud by pivot dělil soubor na zhruba stejně velké části, bude, podobně jako u merge sortu, částí řádově $\log_2(n)$. Celková časová složitost je tedy $O(n \log(n))$.

Poslední částí je dotřídění krátkých úseků.

Časová složitost Quick sortu

Nyní se budeme věnovat rozboru časové složitosti quicksortu. Projdeme první část algoritmu. Všimněte si, že tato část má počet operací úměrný velikosti vstupu. Má tedy časovou složitost $O(n)$.

5	2	7	4	3	8	17	11	9	12	10

Projdeme druhou část algoritmu. Všimněte si, že i tato část má počet operací úměrný velikosti vstupu, a tedy časovou složitost $O(n)$.

						$\leftarrow j$	$i \rightarrow$			
2	3	7	4	5	8	9	10	17	12	11

Kolik bude takových částí? Pokud by pivot dělil soubor na zhruba stejně velké části, bude, podobně jako u merge sortu, částí řádově $\log_2(n)$. Celková časová složitost je tedy $O(n \log(n))$.

Poslední částí je dotřídění krátkých úseků.

Časová složitost Quick sortu

Nyní se budeme věnovat rozboru časové složitosti quicksortu. Projdeme první část algoritmu. Všimněte si, že tato část má počet operací úměrný velikosti vstupu. Má tedy časovou složitost $O(n)$.

5 2 7 4 3 8 17 11 9 12 10

Projdeme druhou část algoritmu. Všimněte si, že i tato část má počet operací úměrný velikosti vstupu, a tedy časovou složitost $O(n)$.

2 3 7 4 5 8 9 10 17 12 11

Kolik bude takových částí? Pokud by pivot dělil soubor na zhruba stejně velké části, bude, podobně jako u merge sortu, částí řádově $\log_2(n)$. Celková časová složitost je tedy $O(n \log(n))$.

Poslední částí je dotřídění krátkých úseků.

Časová složitost Quick sortu

Nyní se budeme věnovat rozboru časové složitosti quicksortu. Projdeme první část algoritmu. Všimněte si, že tato část má počet operací úměrný velikosti vstupu. Má tedy časovou složitost $O(n)$.

5 2 7 4 3 8 17 11 9 12 10

Projdeme druhou část algoritmu. Všimněte si, že i tato část má počet operací úměrný velikosti vstupu, a tedy časovou složitost $O(n)$.

2 3 7 4 5 8 9 10 17 12 11

Kolik bude takových částí? Pokud by pivot dělil soubor na zhruba stejně velké části, bude, podobně jako u merge sortu, částí řádově $\log_2(n)$. Celková časová složitost je tedy $O(n \log(n))$.

Poslední částí je dotřídění krátkých úseků.

Časová složitost Quick sortu

Nyní se budeme věnovat rozboru časové složitosti quicksortu. Projdeme první část algoritmu. Všimněte si, že tato část má počet operací úměrný velikosti vstupu. Má tedy časovou složitost $O(n)$.

5 2 7 4 3 8 17 11 9 12 10

Projdeme druhou část algoritmu. Všimněte si, že i tato část má počet operací úměrný velikosti vstupu, a tedy časovou složitost $O(n)$.

2 3 7 4 5 8 9 10 17 12 11

Kolik bude takových částí? Pokud by pivot dělil soubor na zhruba stejně velké části, bude, podobně jako u merge sortu, částí řádově $\log_2(n)$. Celková časová složitost je tedy $O(n \log(n))$.

Poslední částí je dotřídění krátkých úseků.

Časová složitost Quick sortu

Nyní se budeme věnovat rozboru časové složitosti quicksortu. Projdeme první část algoritmu. Všimněte si, že tato část má počet operací úměrný velikosti vstupu. Má tedy časovou složitost $O(n)$.

5 2 7 4 3 8 17 11 9 12 10

Projdeme druhou část algoritmu. Všimněte si, že i tato část má počet operací úměrný velikosti vstupu, a tedy časovou složitost $O(n)$.

2 3 7 4 5 8 9 10 17 12 11

Kolik bude takových částí? Pokud by pivot dělil soubor na zhruba stejně velké části, bude, podobně jako u merge sortu, částí řádově $\log_2(n)$. Celková časová složitost je tedy $O(n \log(n))$.

Poslední částí je dotřídění krátkých úseků.

Časová složitost Quick sortu

Nyní se budeme věnovat rozboru časové složitosti quicksortu. Projdeme první část algoritmu. Všimněte si, že tato část má počet operací úměrný velikosti vstupu. Má tedy časovou složitost $O(n)$.

5 2 7 4 3 8 17 11 9 12 10

Projdeme druhou část algoritmu. Všimněte si, že i tato část má počet operací úměrný velikosti vstupu, a tedy časovou složitost $O(n)$.

2 3 7 4 5 8 9 10 17 12 11

Kolik bude takových částí? Pokud by pivot dělil soubor na zhruba stejně velké části, bude, podobně jako u merge sortu, částí řádově $\log_2(n)$. Celková časová složitost je tedy $O(n \log(n))$.

Poslední částí je dotřídění krátkých úseků.

Časová složitost Quick sortu

Nyní se budeme věnovat rozboru časové složitosti quicksortu. Projdeme první část algoritmu. Všimněte si, že tato část má počet operací úměrný velikosti vstupu. Má tedy časovou složitost $O(n)$.

5 2 7 4 3 8 17 11 9 12 10

Projdeme druhou část algoritmu. Všimněte si, že i tato část má počet operací úměrný velikosti vstupu, a tedy časovou složitost $O(n)$.

2 3 7 4 5 8 9 10 17 12 11

Kolik bude takových částí? Pokud by pivot dělil soubor na zhruba stejně velké části, bude, podobně jako u merge sortu, částí řádově $\log_2(n)$. Celková časová složitost je tedy $O(n \log(n))$.

Poslední částí je dotřídění krátkých úseků.

Nejhorší případ – Quick sort již setříděného souboru

Úkol: Rozeberte běh algoritmu Quick sort na seřazeném souboru a jeho časovou složitost.

Při rozboru si všimněte, že není pravda, že pivot rozdělí soubor na přibližně stejně velké části.

V každém kroku pivot rozdělí soubor velikosti n na části velikosti nula a $n - 1$. A toto dělení způsobí kvadratickou časovou složitost.

Nejhorší případ – Quick sort již setříděného souboru

Úkol: Rozeberte běh algoritmu Quick sort na seřazeném souboru a jeho časovou složitost.

Při rozboru si všimněte, že není pravda, že pivot rozdělí soubor na přibližně stejně velké části.

V každém kroku pivot rozdělí soubor velikosti n na části velikosti nula a $n - 1$. A toto dělení způsobí kvadratickou časovou složitost.

Nejhorší případ – Quick sort již setříděného souboru

Úkol: Rozeberte běh algoritmu Quick sort na seřazeném souboru a jeho časovou složitost.

Při rozboru si všimněte, že není pravda, že pivot rozdělí soubor na přibližně stejně velké části.

V každém kroku pivot rozdělí soubor velikosti n na části velikosti nula a $n - 1$. A toto dělení způsobí kvadratickou časovou složitost.