In-place Mergesort

Dejdar Jan

8. 11. 2016

1 Definice problému

Jednou z nevýhod populárního třídícího algoritmu merge sort je paměťová složitost $O(n \log n)$ u neoptimalizované verze, případně $O(n)$ u implementace, která si pracovní pole alokuje dopředu. Existuje však i in-place verze tohoto algoritmu, kterou se zde budu zabývat.

Efektivitu algoritmu se pokusím zvýšit nahrazením in-place verze algoritmu algoritmu merge sortem a insertion sort pro malá pole. Dále budu zkoumat vliv rozložení vstupní posloupnosti. Změřím časy jednotlivých verzí algoritmu pro náhodnou, seřazenou a obráceně seřazenou vstupní posloupnost.

1.1 Popis sekvenčního algoritmu

Klasický merge sort používá přídavné pomocné pole pro slévaní dvouseřazených částí pole. V implementaci in-place verze budeme muset použít zbytek původního pole jako pracovní prostor pro toto slévání. Prvky v tomto zbytku pole ale není možné přepsat, protože je budeme řadit později. Myšlenka jak toho docílit spočívá v tom, že když chceme menší z dvou právě porovnávaných prvků umístit do pracovního prostoru, vyměníme tento prvek s příslušným prvkem v pracovním prostoru. Po skončení slévání tedy dvě původně seřazená podpole obsahují prvky, které předtím byly v pracovním prostoru, zatímco tento obsahuje seřazené pole.

Při slévání musí být splněny dvě podmínky:

1. Pracovní prostor musí být dostatečně velký, aby pojal obě části pole, které chceme slévat.

2. Pracovní prostor se může překrývat se seřazenými podpoli, ale je nutné zajistit, že nepřepíšeme žádné neslité prvky.

S výše algoritmem je triviální sestrojit algoritmus, který seřadí polovinu původního pole, protože máme dostatek pracovního prostoru pro slévání seřazených prvků. Otázkou zůstává, jak seřadit zbývající neseřazenou polovinu pole.

| ... SEŘAZENÉ ... | ... NESEŘAZENÉ ... |

Tabulka 1: Seřazená polovina pole

Intuitivní postup by byl rekurzivně seřadit druhou polovinu neseřazeného pole, takže by zbyla jen $\frac{1}{4}$ neseřazeného pole. Problém je v tom, že musíme slít B a A, na což nemáme dostatečně velký pracovní prostor.

| ... NESEŘAZENÁ $\frac{1}{4}$ ... | ... SEŘAZENÁ $\frac{1}{4}$ (B) ... | ... SEŘAZENÁ $\frac{1}{2}$ (A) ... |

Tabulka 2: Na slití A a B nemáme dostatečný pracovní prostor

Klíčová myšlenka spočívá v tom, že místo abychom seřadili druhou polovinu neseřazeného pole, seřadíme tu první, čímž umístíme pracovní prostor mezi dvě seřazené části pole. Pracovní prostor se pak bude překrývat se seřazenou polovinou pole A.

Nyní uvažujme dva extrémní případy, které mohou nastat:

1. Všechny prvky v B jsou menší než prvky v A. Potom slívací algoritmus postupně přesune B do pracovního prostoru. Protože jejich velikosti jsou stejné, vše je v opřádku.

2. Všechny prvky v B jsou větší než v A. V tomto případě slívací algoritmus postupně přesouvá prvky z A do pracovního prostoru. Po zaplnění celého původního pracovního prostoru začne algoritmus přepisovat první polovinu pole A. Nejedná se ale o neslité prvky, takže podle druhé podmínky je vše v pořádku. Pracovní pole se tak tedy přesune na pravou stranu. Nyní algoritmus začne prohazovat prvky z B s pracovním prostorem. Pracovní pole se tak tedy přesune na levou stranu, viz tabulku.

| ... SEŘAZENÁ $\frac{1}{4}$ (B) ... | ... PRACOVNÍ PROSTOR $\frac{1}{4}$ ... | ... SEŘAZENÁ $\frac{1}{2}$ (A) ... |

$\Downarrow$

| ... PRACOVNÍ PROSTOR $\frac{1}{4}$ (B) ... | ... SLITÉ $\frac{3}{4}$ ... |

Tabulka 3: Slití A a B

Takto algoritmus pokračuje a rekurzivně zmenšuje a postupně slévá seřazená pole. Když se velikost pole zmenší pod určitou mez, můžeme přepnout na jiný řadící algoritmus (standardní merge sort, Insertion sort...) a tím se pokusit optimalizovat rychlost výsledného algoritmu.