Слоест вектор и DoS върху AVL дърво

Мария Гроздева

Навсякъде в текста с **n** ще бележим броя на елементите в съответната структура.

Слоест вектор (Tiered vector) - структура от данни, която поддържа следните операции:

- **elementAtRank**(r) - извличане на елемент по индекс (сложност - **O(1)**);
- **insertAt**(r, el) - вмъкване на елемента el на индекс r (сложност - **O(sqrt(n))**);
- **removeAt**(r) - премахване на елемента, намиращ се на индекс r (сложност - **O(sqrt(n))**);
- **pushBack**(el) - добавяне в края на елемента el (сложност - **O(1)**);
- **pushFront**(el) - добавяне в началото на елемента el (сложност - **O(1)**);
- **popBack**() - премахване на последния елемент (сложност - **O(1)**);
- **popFront**() - премахване на първия елемент (сложност - **O(1)**)

DoS върху AVL дърво - структура от данни, която поддържа следните операции:

- **elementAtRank**(r) - извличане на елемент по индекс (сложност - **O(log(n))**);
- **insertAt**(r, el) - вмъкване на елемента el на индекс r (сложност - **O(log(n))**);
- **removeAt**(r) - премахване на елемента, намиращ се на индекс r (сложност - **O(log(n))**);
- **pushBack** (el) - добавяне в края на елемента el (сложност - **O(log(n))**);
- **pushFront**(el) - добавяне в началото на елемента el (сложност - **O(log(n))**);
- **popBack**() - премахване на последния елемент (сложност - **O(log(n))**);
- **popFront**() - премахване на първия елемент (сложност - **O(log(n))**)

Сравнение на производителността на двете структури

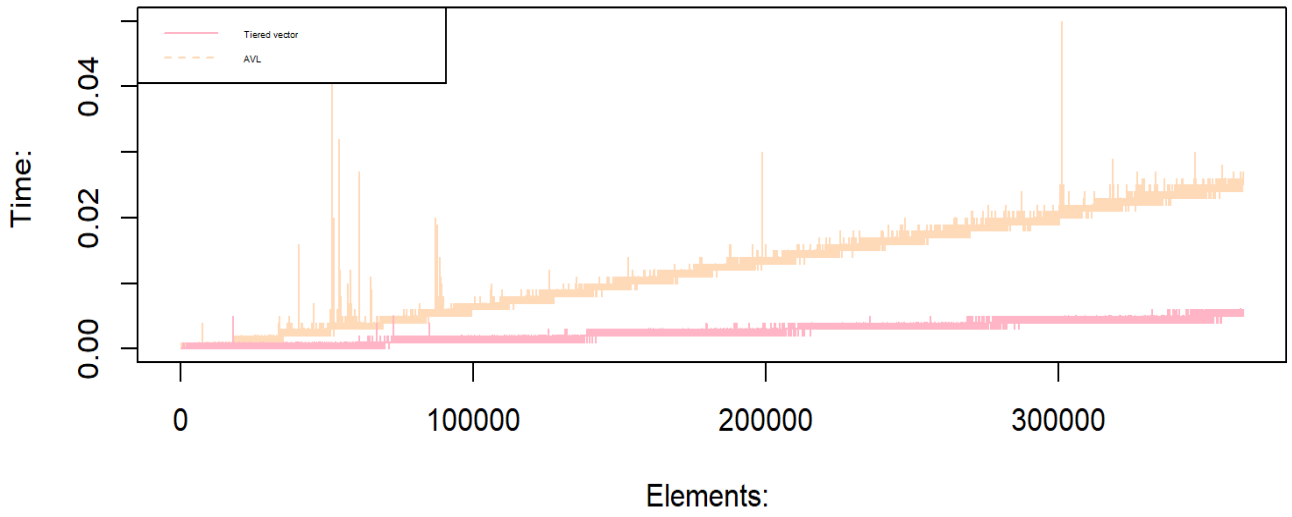
В този раздел се прави анализ на операциите, поддържани от двете структури. Анализът се състои в провеждане на различни

експерименти и сравняване на емпиричните сложности на DoS върху AVL и Tiered vector.

- ## Извличане на елемент по индекс:

При този експеримент се индексират **всички** елементи в структурата.

Сложността на тази операция е O(log(n)) при DoS върху AVL дърво и O(1) при слоест вектор. От графиката виждаме, че емпиричната сложност съвпада с теоретичната.

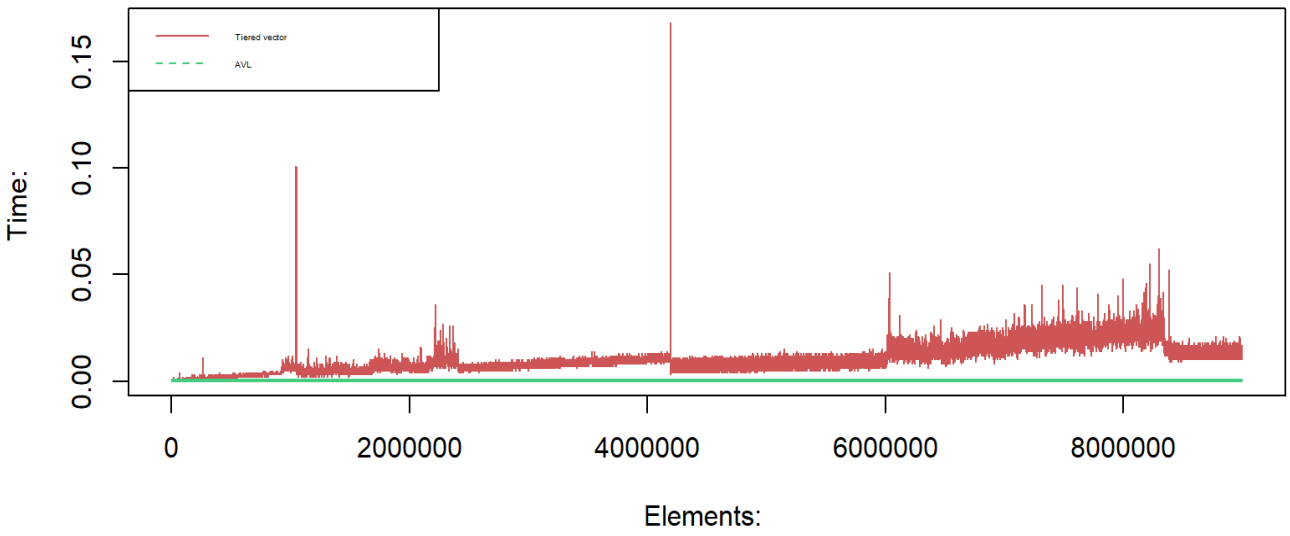


- ## Вмъкване на елемент на произволна позиция:

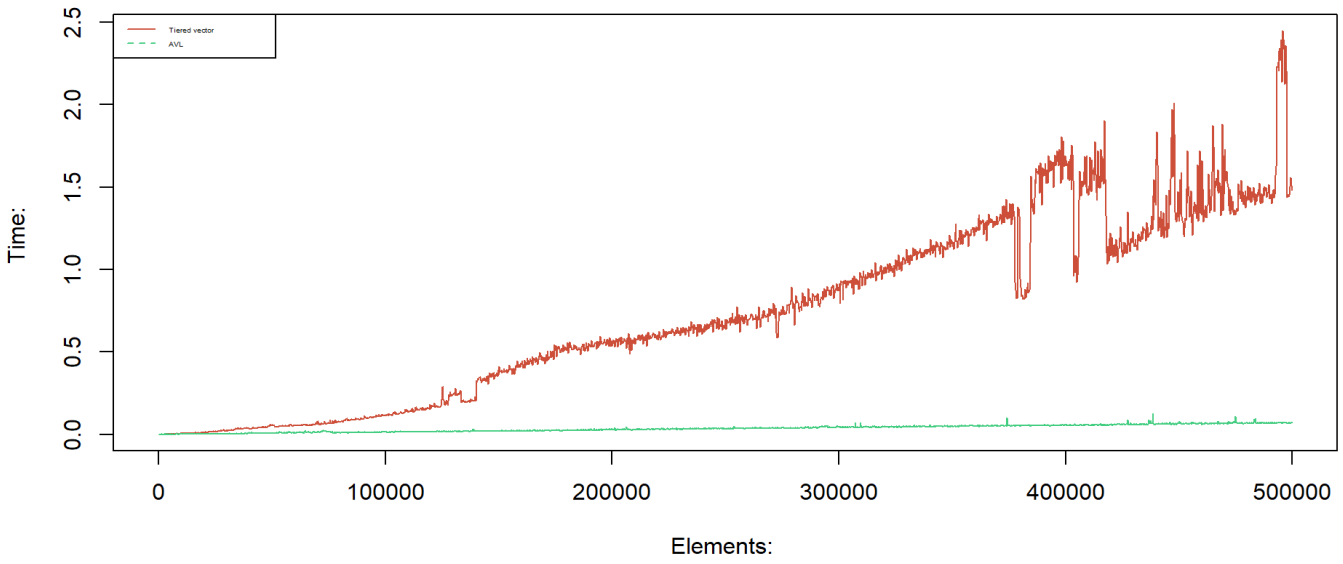
Проведени са два различни експеримента - при първия се вмъкват константен брой (независещ от текущата големина на структурата) елементи, а при втория - n/2 елемента при налични n.

Сложността на тази операция е O(log(n)) при DoS върху AVL дърво и O(sqrt(n)) при слоест вектор.

1. **Първи експеримент** - при n елемента в структурата се добавят нови 500. Резултатите, които наблюдаваме, са много близки до теоретичните (можем да го видим като разгледаме графиките на функциите log(n) и sqrt(n)).



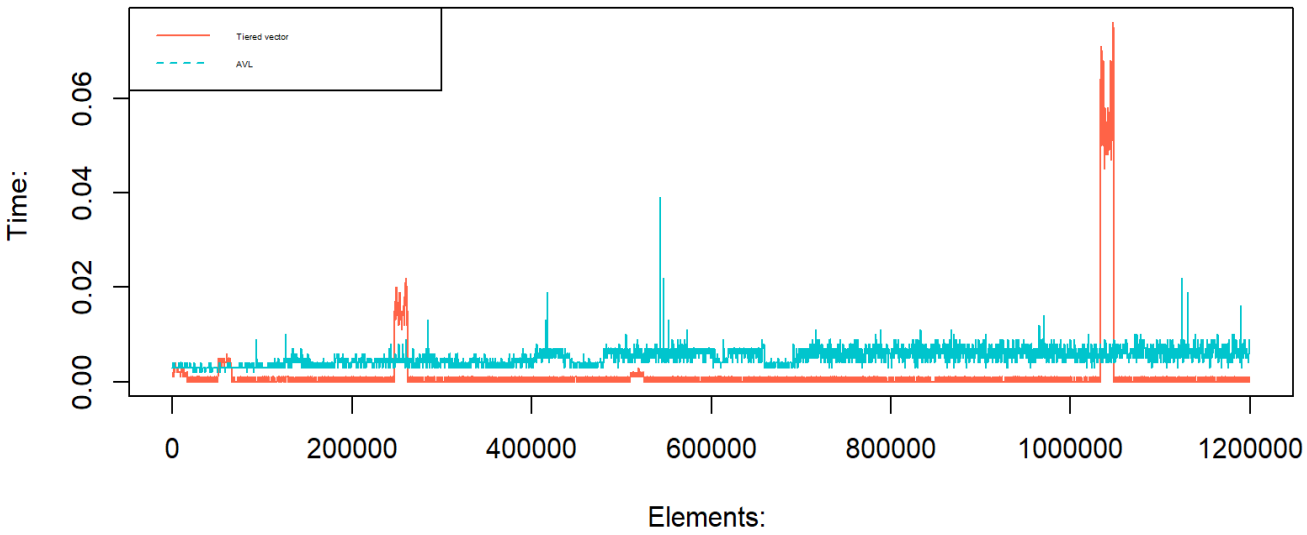
2. **Втори експеримент** - при n елемента в структурата се добавят нови n/2. Сравнено с горния експеримент, разликите при този са значително изострени. Това е очакваното поведение, тъй като при този експеримент с нарастването на n нараства и броят на добавяните елементи.



- # Добавяне на елемент в края:

При този експеримент за всяко n се добавят 15 000 елементи в края.

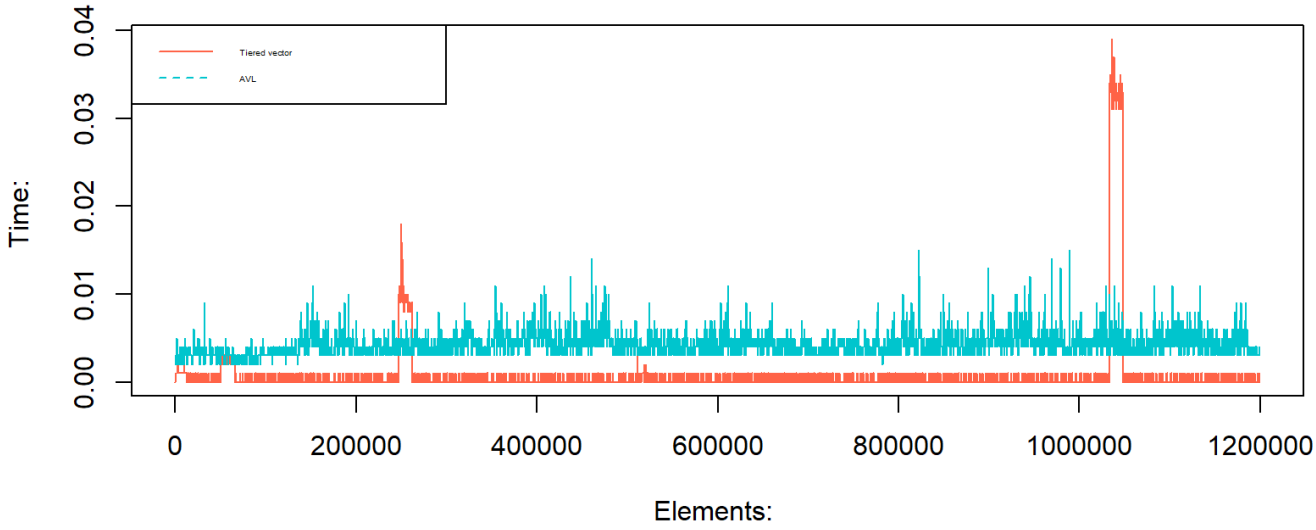
Сложността на тази операция е O(log(n)) при DoS върху AVL дърво и O(1) при слоест вектор. Отново емпиричните данни съответстват на теоретичните резултати.



- **Добавяне на елемент в началото:**

При този експеримент за всяко n се добавят 15 000 елементи в началото.

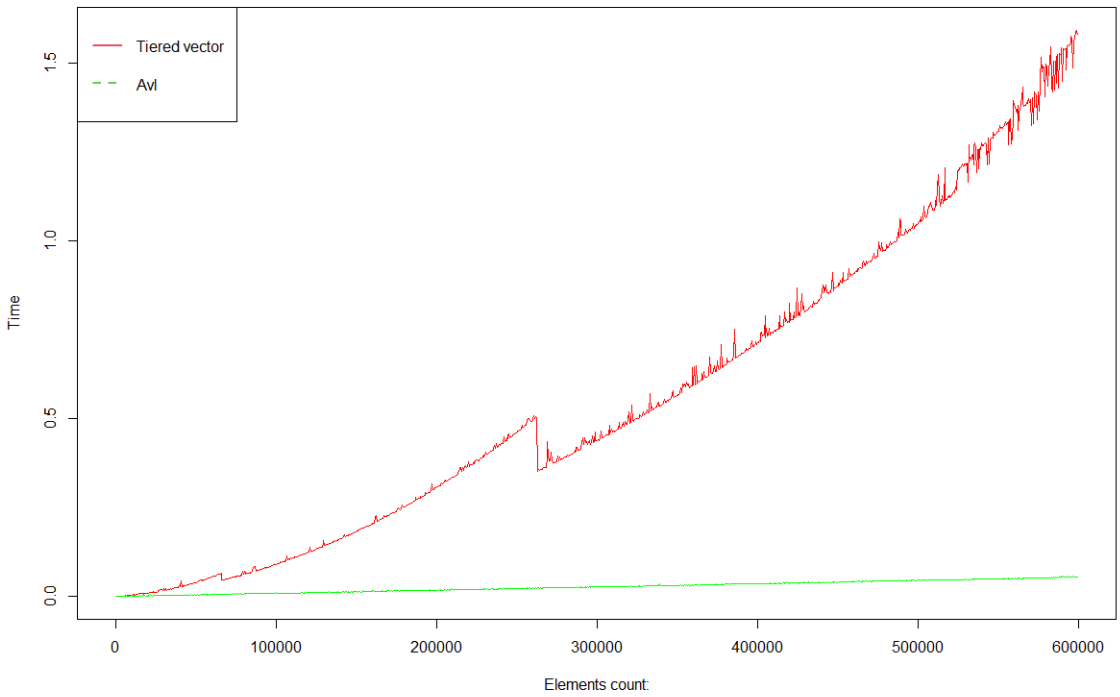
Сложността на тази операция е O(log(n)) при DoS върху AVL дърво и O(1) при слоест вектор. Очакваме получените резултати да бъдат много сходни на тези от горния експеримент.



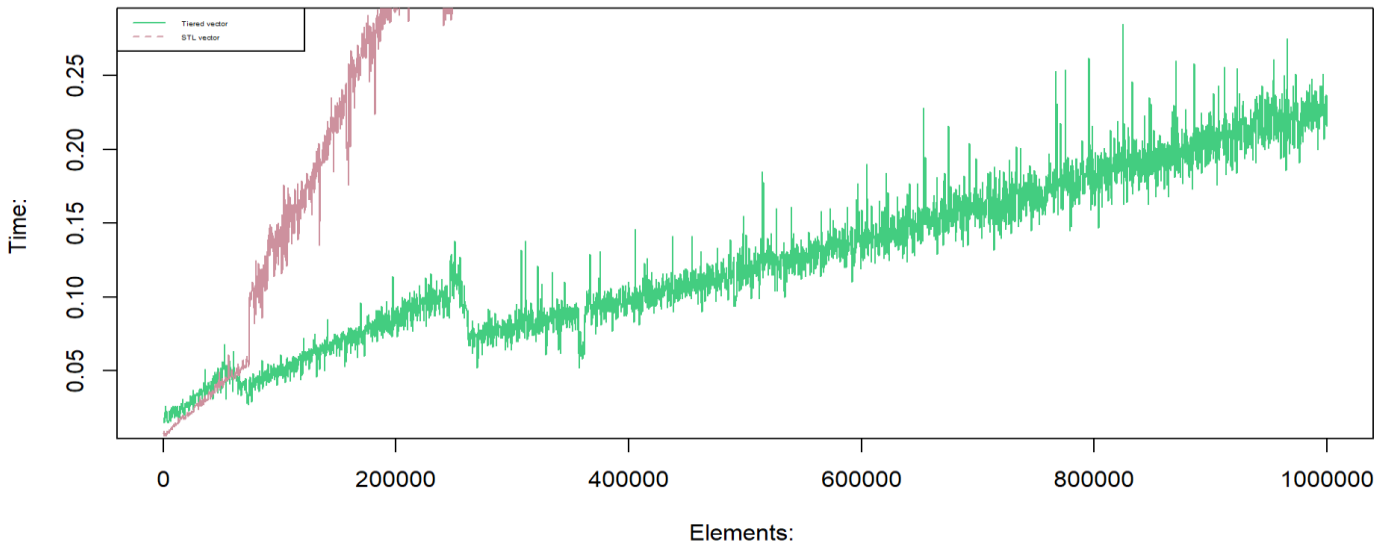
- **Измъкване на елемент от произволна позиция:**

При n елемента в структурата се премахват половината. И тук, както при добавянето, когато броят елементи, които премахваме, зависи от големината на входа, разликите в производителността на двете структури са значително изострени.

Сложността на тази операция е O(log(n)) при DoS върху AVL дърво и O(sqrt(n)) при слоест вектор. Въпреки че в асимптотичния смисъл тези сложности са същите като при вмъкване на елемент, измъкването при AVL е теоретично по- бавно. Това се дължи на факта, че при добавяне в дървото се извършват най- много 2 ротации, докато при премахване могат да се извършат до log(n) ротации. Въпреки това, при наблюдаваните експериментални данни поведението на дървото при добавяне и при премахване на елементи е много сходно.



- **Tiered vector vs Vector in C++ STL - вмъкване на произволна позиция:**



Код от експериментите:

```cpp
for (size_t i = 250; i < 1000000; i += 250)
{
    TieredVector<int> v;
    addToTV(v, i); // inserts the elements from 0 to i

    clock_t begin = clock();
    for (size_t j = 0; j < 15000; j++)
        v.insertAt(v.getSize() / 2, 100);
    clock_t end = clock();

    double elapsed_secs = double(end - begin) / CLOCKS_PER_SEC;
}
```

```cpp
for (size_t i = 250; i < 1000000; i += 250)
{
    std::vector<int> v(i, 100); // inserts 100 i times
    auto it = v.begin();

    clock_t begin = clock();
    for (size_t j = 0; j < 15000; j++)
        v.insert(it + v.size() / 2, 999);
    clock_t end = clock();

    double elapsed_secs = double(end - begin) / CLOCKS_PER_SEC;
}
```