За дървото на вземане на решение (*Decision Tree*) при Insertion Sort

Минко Марков

27 октомври 2011 г.

```
Insertion Sort(A[1, ..., n])
1  for i ← 2 to n
2      key ← A[i]
3      j ← i − 1
4      while j > 0 and key < A[j] do
5          A[j + 1] ← A[j]
6          j ← j − 1
7      A[j + 1] ← key
```

Да си представим работата на Insertion Sort върху вход с големина три. Нека входът е

$$a_1, a_2, a_3$$

В следващите разсъждения, с малки букви, примерно "a_1", означаваме елементите от входа, а с големи букви, примерно "$A[1]$", означаваме елементите от масива $A[]$ в момента. Тоест, в самото начало $A[] = [a_1, a_2, a_3]$, но след няколко стъпки от изпълнението може, примерно, $A[] = [a_2, a_3, a_1]$. Допускаме, че елементите от входа са два по два различни. Точно едно от следните е вярно:

$$a_1 < a_2 < a_3 \tag{1}$$
$$a_1 < a_3 < a_2 \tag{2}$$
$$a_2 < a_1 < a_3 \tag{3}$$
$$a_2 < a_3 < a_1 \tag{4}$$
$$a_3 < a_1 < a_2 \tag{5}$$
$$a_3 < a_2 < a_1 \tag{6}$$

Ще наречем тези възможности, *пермутациите*. Да сортираме входа е същото като да определим, кое от (1), . . . , (6) е вярно, тоест коя от пермутациите (на елементите на входа) реализира сортиране. Insertion Sort е сортиращ алгоритъм, базиран на директни сравнения. Сравненията стават на едно единствено място: на ред 4, оцветеното каре $\text{key} < A[j]$. Първото сравнение, което прави алгоритъмът независимо от това какъв е входът, е $a_2 < a_1$ – изразено чрез това, кои елементи от входа биват сравнявани.

I Ако $a_2 < a_1$ е истина, то точно тези три пермутации остават възможни:

$$a_2 < a_1 < a_3$$
$$a_2 < a_3 < a_1$$
$$a_3 < a_2 < a_1$$

Тогава алгоритъмът променя $A[\,]$ в текущото изпълнение на **for**-цикъла така, че $A[\,] = [a_2, a_1, a_3]$, без да прави повече сравнения на елементи на ред 4, и продължава със следващата итерация на **for**-цикъла.

II Ако $a_2 < a_1$ не е истина, то точно тези три пермутации остават възможни:

$$a_1 < a_2 < a_3$$
$$a_1 < a_3 < a_2$$
$$a_3 < a_1 < a_2$$

Тогава алгоритъмът не променя $A[\,]$ в текущото изпълнение на **for**-цикъла, така че $A[\,] = [a_1, a_2, a_3]$ и, без да прави повече сравнения на елементи на ред 4, продължава със следващата итерация на **for**-цикъла.

В случай, че $a_2 < a_1$, следващото сравнение на входни елементи е $a_3 < a_1$. Това се вижда лесно, ако си спомним, че текущият $A[\,]$ е $A[\,] = [a_2, a_1, a_3]$.

I.1 Ако $a_3 < a_1$ е истина, то точно тези две пермутации остават възможни:

$$a_2 < a_3 < a_1$$
$$a_3 < a_2 < a_1$$

Текущият $A[\,]$ става $A[\,] = [a_2, a_1, a_1]$. В момента a_3 се съхранява в променливата key. Следващото сравнение, което се върши, е $a_3 < a_2$.

I.1.a Ако $a_3 < a_2$ е истина, то точно една пермутация остава възможна:

$$a_3 < a_2 < a_1$$

Що се отнася до изпълнението на алгоритъма, то **while**-цикълът се изпълнява още (точно) веднъж, като $A[]$ става $A[] = [a_2, a_2, a_1]$. След това a_3 се слага на първа позиция в $A[]$ на ред 7 и $A[]$ става $A[] = [a_3, a_2, a_1]$. Следва краят на алгоритъма.

I.1.b Ако $a_3 < a_2$ не е истина, то точно една пермутация остава възможна:

$$a_2 < a_3 < a_1$$

Що се отнася до изпълнението на алгоритъма, то **while**-цикълът повече не се изпълнява. a_3 се слага на втора позиция в $A[]$ на ред 7 и $A[]$ става $A[] = [a_2, a_3, a_1]$.

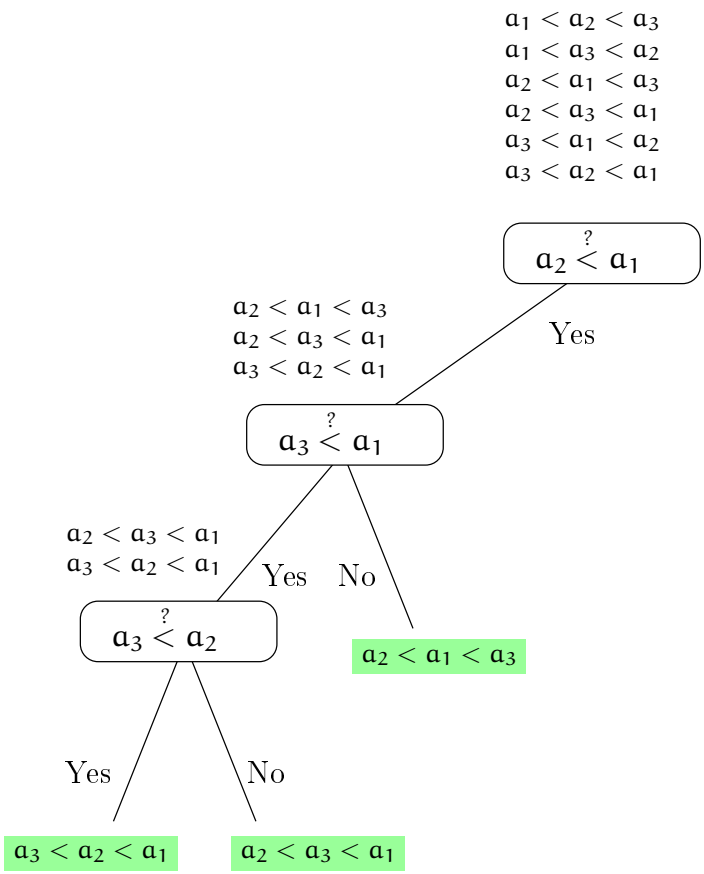
I.2 Ако $a_3 < a_1$ не е истина, то точно една пермутация остава възможна:

$$a_2 < a_1 < a_3$$

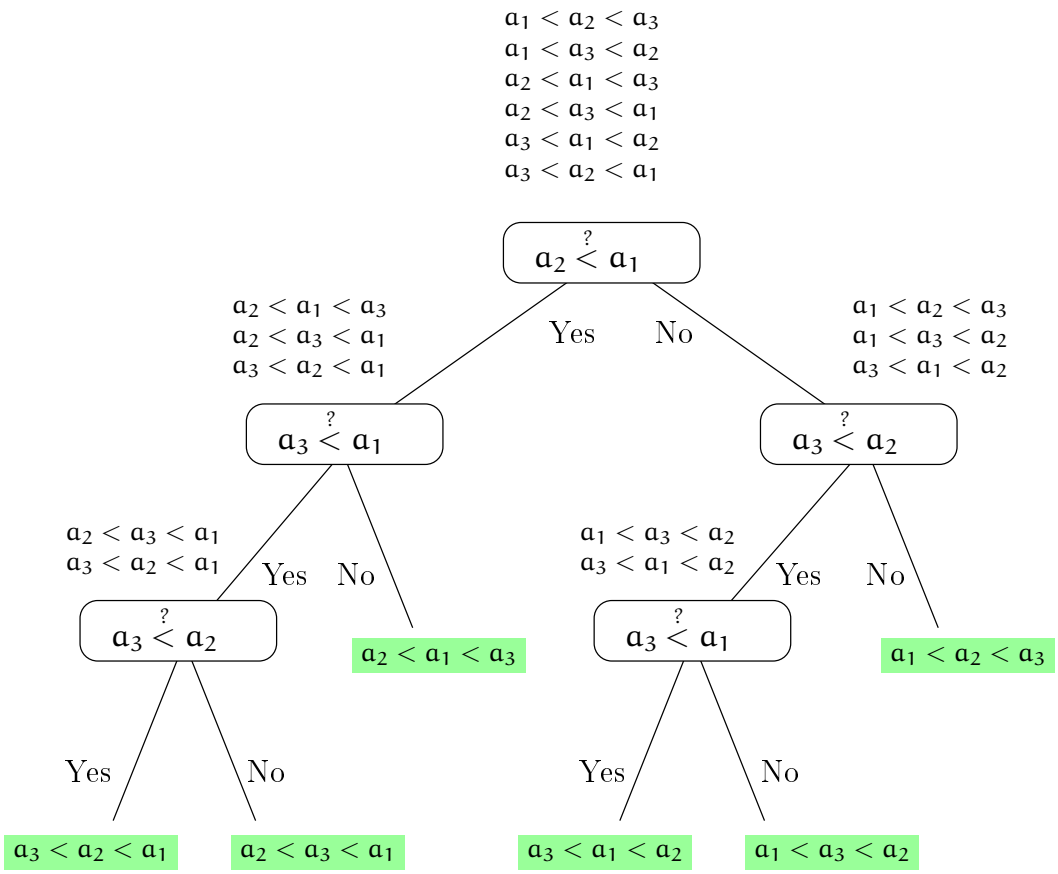
Що се отнася до изпълнението на алгоритъма, то в този подслучай това е краят. Повече сравния не се правят и масивът остава $A[] = [a_2, a_1, a_3]$.

Четирите подслучая на случай **I** можем да опишем синтезирано в дървовидната структрура, показана на Фигура 1. Върховете са два вида – върхове-сравнения, в които се задава въпрос от вида $a_i \overset{?}{<} a_j$, и листа, оцветени в зелено. Над всеки връх-сравнение записваме пермутациите, които са възможни преди това сравнение (тоест, консистентни с отговорите на досега отговорените въпроси). Естествено, над корена са шестте пермутации, възможни преди първото сравнение. Листата са множествата от възможни пермутации, състоящи се само от една пермутация. Листата съответстват на възможните изходи на алгоритъма. Ако извършим аналогичното изследване и на подслучаите на случай **II**, ще получим структурата, показана на Фигура 2. Дървото е двоично, понеже всеки от въпросите има точно два възможни отговора – Yes и No.

Очевидно е възможно да мислим за тази дървовидна структура от въпроси и възможни отговори като за нещо първично. С други думи, ние я изведохме, изследвайки напълно различната работа на алгоритъма върху шестте различни възможни входа с големина три; но можем да си мислим за тази структура не като за изведена от алгоритъма, а като за схема от въпроси и отговори, съгласно която можем да решим коя от шестте възможни пермутации е сортираната. Това, че има листа на разстояние две и на разстояние три от корена означава, че съгласно тази схема, при някои входове можем само с два въпроса да решим коя е



Фигура 1: Подслучаите, когато $a_2 < a_1$ е истина.



Фигура 2: Пълното дърво на вземане на решения.

сортираната пермутация, докато при други входове се налага да зададем три въпроса. Броят на въпросите за дадена пермутация, естествено, е броят на не-листата по пътя между нея и корена, тоест дължината на пътя. Височината на дървото тогава е максималният брой въпроси, които биват зададени от тази схема.

Очевидно съществуват и други схеми от въпроси и отговори за сортиране на вход с големина три. Те съответстват на други сортиращи алгоритми. Забележете, че ***всяка*** схема задава в най-лошия случай поне три въпроса. Да допуснем противното – съществува схема, която с не повече от два въпроса от вида $a_i \overset{?}{<} a_j$ може да реши коя е сортираната пермутация (при вход с големина три). Тогава съответното дърво има не повече от четири листа, тъй като дървото е двоично и с височина две. Това означава, че схемата може да различава не повече от четири възможни пермутации. А, както знаем, има шест пермутации на три елемента и ***всяка*** схема от въпроси и отговори, която изчислява сортираната пермутация, трябва да може да ги различава. Тоест, дървото трябва да има поне шест листà.