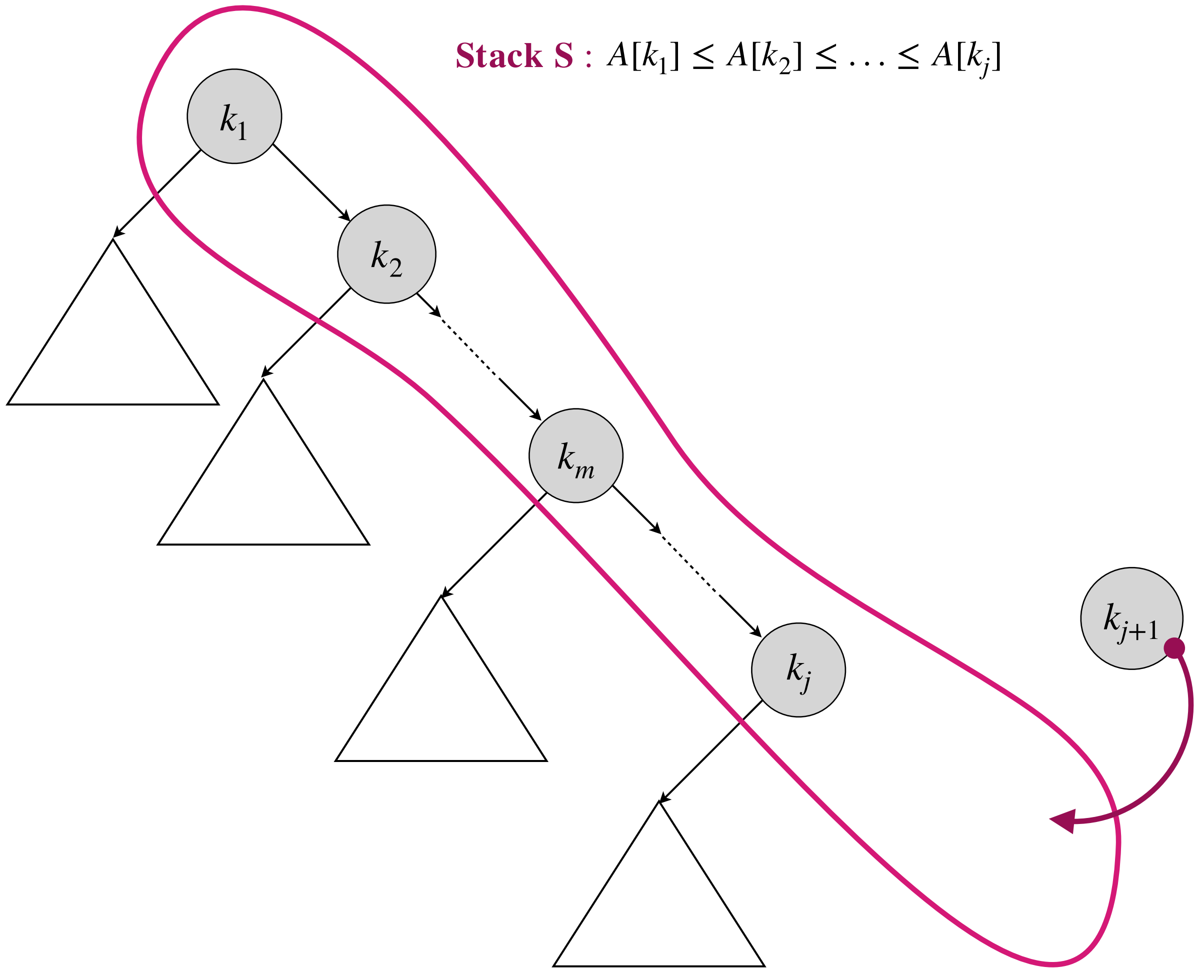
Построяване на декартово дърво (алтернативен подход)

Ще създадем декартовото дърво като родителски масив $parent$ (който лесно може да се преобразува в списък на съседства или каквато и да е друга форма на представяне). Тъй като върховете на декартовото дърво са уникални индекси, то в масива $parent$ ще използваме индекса за връх, а стойността в този индекс за да покажем кой индекс е баща му.

Идея на алгоритъма:
Ще използваме стек, в който ще слагаме върховете от „*най-десния път*“ (под-най десен път ще разбираме най-десния клон на дървото). Той ще се запълва всеки път когато пристига връх, зад чиито индекс стои по-ниска стойност от последната в стека. Всеки път когато дойде индекс с по-висока стойност от последната в стека – ще започваме да вадим елементи от стека, докато не срещнем по ниска и тогава ще вкараме текущата стойност в стека. Стека ще ни даде възможност лесно да съхраняваме информация, чрез която бързо да проверяваме на кой елемент искаме да прикачваме роднински връзки.

Псевдо код:

procedure $Build()$ {
  $parent[0...n-1]$ // създаваме родителският масив, който ще върнем
  $stack\ S \leftarrow \emptyset$ // създаваме празен стек, който да съхранява текущия най-десен път
  $last \leftarrow -1$
  for $i \leftarrow 1$ **to** $n-1$ **do**
    while $S \neq \emptyset$ **and** $A[S.lookup()] \geq A[i]$ **do** // запазваме върха, чиито баща ще падне в ляво от най-десния път (защото на негово място ще дойде текущия връх (индекс))
      $last \leftarrow S.lookup()$
      $S.extract()$
    done
    if $S \neq \emptyset$ **then** // закачаме текущия връх към върха в края на вече актуализирания стек
      $parent[i] \leftarrow S.lookup()$
    if $last \geq 0$ **then**
      $parent[last] \leftarrow i$
    $S.insert(i)$
  done
  return $parent$
}



Всеки път когато дойде следващ връх (индекс), например k_{j+1}, той ще търси къде да се „покатери" по най-десния път и да предположим, че това ще е върха k_m, за чиито баща не е изпълнено условието $A[k_{m-1}] \not\geq A[k_{j+1}]$, $\big($ т.е. $A[k_{m-1}] < A[k_{j+1}]\big)$.Тогава на мястото на бащата на k_m:

$k_{m-1} = parent[k_m]$ ще бъде новопоставения връх, а на него ще прикачваме новите роднински връзки.

<u>Заключение</u>:
Алгоритъма се базира на една много стара идея, а именно - намиране на най-дълга растяща подредица в редица от числа за линейно време. Там по аналогичен начин използвайки стек детерминираме тази редица като връщаме началния ѝ индекс (и дължината ѝ).
Тук този алгоритъм не е нещо по-различно на идеино ниво от този, който разгледахме с помощната функция $Insert$, но единственото предимство което се сещам е, че може би доказателството за линейната сложност на този алгоритъм ще е доста тривиално: всеки връх влиза точно веднъж в стека и след като излезе – повече никога не се връща (това е така, защото веднъж кривнал в ляво от най-десния път връх, повече няма как да е част от този път). Накрая в стека ще остане съхранен целия финален най-десен път.