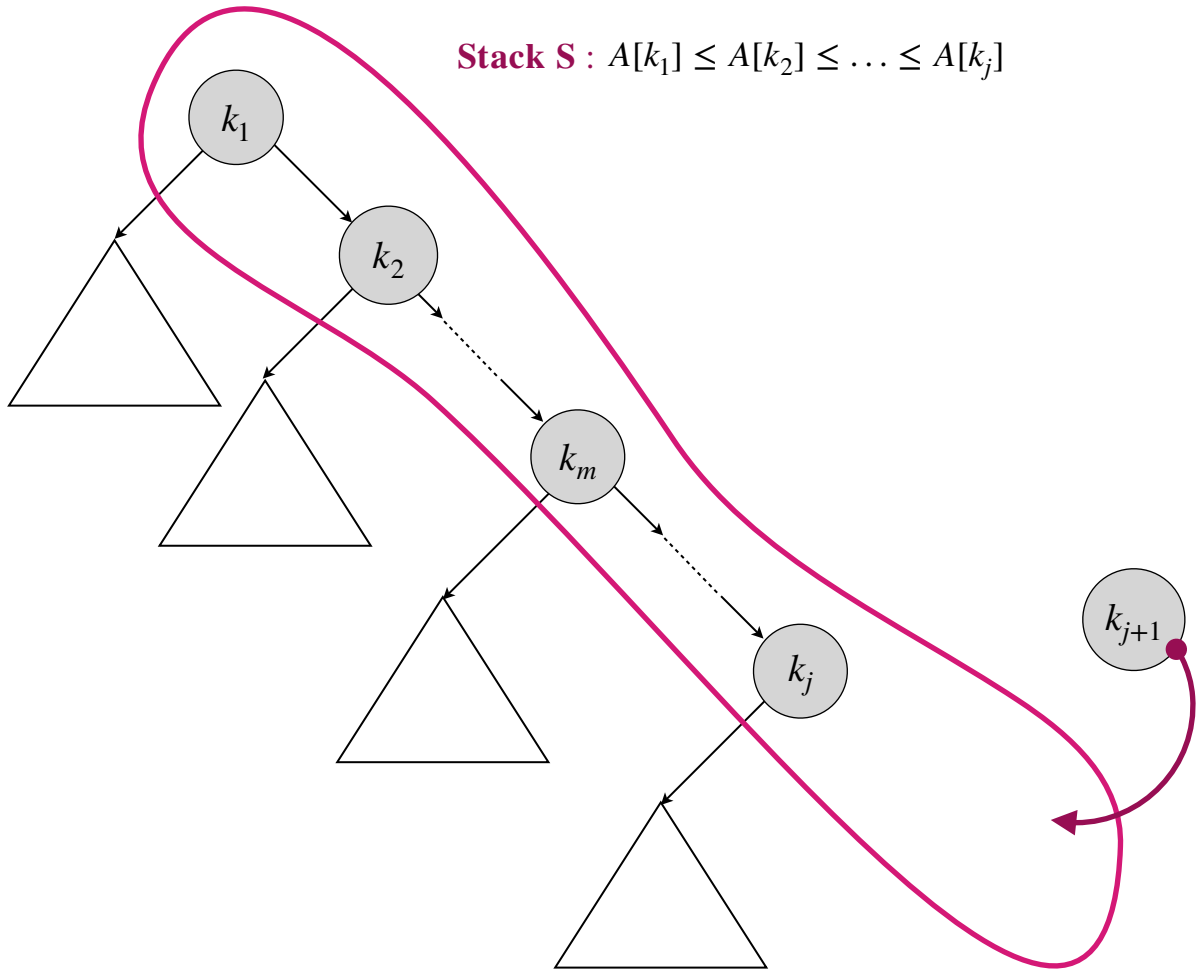
Построяване на декартово дърво (алтернативен подход)

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*“. Той ще се запълва всеки път когато пристига връх, зад чиито индекс стои по-ниска стойност от последната в масива. Всеки път когато дойде индекс с по-висока стойност в масива 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 родителския масив, който ще върнем
  stack S ← Ø      //създаваме празен стек, който да репрезентира текущия най-десен път
  for i ← 1 to n − 1 do
    while S ≠ Ø and A[S.lookup()] ≥ A[i] do
      last ← S.lookup() // запазваме върха, чиито баща ще падне в ляво от най-десния
      S.extract()       път (защото на негово място ще дойде текущия връх (индекс))
    done
    if S ≠ Ø
      parent[i] ← S.lookup() // закачаме текущия връх към върха в
    if last ≥ 0              края на вече актуализирания стек
      parent[last] ← i
    S.insert(i)
  done
  return parent
```



Всеки път като дойде следващ връх (индекс), например k_{j+1}, той ще търси къде да се покатери по-най десния път и да речем това ще е върха k_m, за чиито баща не е изпълнено условието $A[k_{m-1}] \not\geq A[k_{j+1}]$, $\left(\text{ т.е. } A[k_{m-1}] < A[k_{j+1}]\right)$.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.

Заключение:
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 като предимство което се сещам е, че може би доказателството за линейната сложност на този алгоритъм със стека ще е доста тривиална: 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 имаме съхранен целия финален най-десен път.