**Soluție – problema s2c**

*Autor: prof. Ionel-Vasile Piț-Rada, Colegiul Național “TRAIAN”, Drobeta Turnu Severin*

Se definesc:

* best[i][j] = lungimea maximă a unui subșir 2-crescător în care primele două elemente sunt a[i] și a[j], 1<=i<j<=n, astfel avem best[i][j]=1+maxim{best[j][k] / i<j<k și a[i]<a[k]}
* L=maxim{best[i][j] / 1<=i<j<=n}
* poz[i][lung] = poziția j pentru care un subșir 2-crescător care pornește cu a[i] și a[j] are lungimea maximă egală cu lung; în cazul în care există mai multe astfel de poziții se va alege acea poziție j pentru care valoarea a[j] este maxim
* value[x][lung] = cea mai mica valoare care poate constitui ultimul element pentru un s2c care are lungimea lung si penultimul element egal cu x.

**Soluție de complexitate O(n3)**

Se calculează best[i][j] și în același timp L.

**Soluție de complexitate O(n2L)**

Se calculează poz[i][lung] și în același timp L

**Soluție de complexitate O(n\*L\*log(n))**

Parcurgem elementele a[i] in ordine. Ne intrebam ce actualizari produce un nou element a[i]. Pentru fiecare lungime lung, putem cauta valoarea min(value[x][lung]) pentru orice x < a[i], fie ea minval, iar apoi putem actualiza value[minval][lung+1] cu min(value[minval][lung+1], a[i]). Pentru a aduce complexitatea la cea descrisa mai sus, vom utiliza un AIB pentru fiecare dimensiune lung care sa suporte query-uri/update-uri si o normalizare a valorilor a[i], pentru ca aceste AIB-uri sa fie de dimensiune N, nu VALMAX.

**Soluție de complexitate O(n2)**

Pentru calcularea poz[i][lung] fixăm j și observăm că pentru i1<i și i2<i și a[i1]<a[i2], avem best[i1][j] >= best[i2][j]. Astfel , dacă vom parcurge **crescător** valorile a[i] cu i<j și **descrescător** după lungimea lung<=L, atunci cele două parcurgeri se pot realiza interclasat în O(n+L).