3.15

a) Per calcular la recurrència observem el següent, si volem (i,j) aleshores pot ser que el subconjunt màxim (widle) contingui una corda (i,k) on $i<k\leq j$ o no. En el cas que la contingui observem que el problema queda dividit en dos subproblemes que han de ser òptims, $(i+1,k-1)$ i $(k+1,j)$, altrament tenim una instància del subproblema òptim $(i+1,j)$

Per tant $T(i,j)=0$ per $i\geq j$ (cas base)

$$T(i,j)=\begin{cases} T(i+1,j) \text{ si no hi ha corda } (i,k) \quad i<k\leq j \\ \max\{T(i+1,j),\ 1+T(i+1,k-1)+T(k+1,j)\} \text{ si tenim corda } (i,k) \ i<k\leq j \end{cases}$$

b) L'algorisme consisteix en implementar la programació dinàmica seguint aquesta recurrència, com cada vèrtex té una corda

. les podem emmagatzemar en un vector de mida $2n$ de forma que $V[i]=.$ si hi (i,j) corda i per tant podem determinar si hi ha una corda de la forma que volem en temps constant.

Aleshores cada càlcul de $T(i,j)$ és constant i tenim $0<i\leq j\leq 2n-1 \Rightarrow$

$\Rightarrow \Theta(n^2)$ posicions. Per tant l'algorisme calcula $T(0,2n-1)$ en temps $\Theta(n^2)$.

Per determinar el conjunt widle en cada pas de l'algorisme ens apuntem

$C(i,j)=\begin{cases} 1 \text{ si pillem corda amb extrem } i \\ 0 \text{ altrament} \end{cases}$