Pipeline Subtask 4

Am Anfang wird ein Struct mit den Attributen counter, looking_from und last definiert.
Dann wird ein M langer Vektor pumps mit jeweils einem dieser Structs an jedem Index erstellt.

Der Server gibt uns die erste Position der ersten Punkte von N Pumpen und dieser Wert wird für jedes Struct des M-langen Vektors als looking_from und last eingestellt mit einem for-loop. Der counter wird auch bei allen um 1 erhöht.

Dann wird über die N Pumpen iteriert, die nicht gespeichert werden müssen, da der Server bei jedem Iterationsschrit die Pumpe N_i geben kann und das reicht.
Folgendes passiert dann in einem For Loop der über pumps iteriert mit j als index:

Wenn der Wert des m_j.counters == -1 ist, dann wird continued.
Wenn der Wert des aktuellen N_i - $pumps_j$.last > j+1 ist, dann wird der m_j.counter auf -1 gesetzt.
Sonst:
Wenn N_i-$pumps_j$.looking_from == j+1 ist, dann wird counter +1 und $pumps_j$.looking_from wird auf N_i gesetzt. Dabei wird auch noch gefragt, ob das das letzte N_i ist, also i == N-1, wenn ja dann wird der Counter auch nochmal erhöht.
Sonst:
Wenn N_i-m_j.looking_from > j+1 ist, dann wird counter ++ und m_j.looking_from auf m_j.last gesetzt. Dabei wird auch noch gefragt, ob das das letzte N_i ist, also i == N-1, wenn ja dann wird der Counter auch nochmal erhöht.

Nach diesen Abfragen wird $pumps_j$.last auf N_i gesetzt.

Wenn diese beiden ineinander verknüpften for loops terminiert sind, ist in $pumps_j$.counter die Anzahl benötigter Pumpen für ihr LTC, dass j+1 ist.

Dieser Algorithmus benötigt O(M) Speicher und O(N*M) Laufzeit weil wir für jedes N_i über M langer Vector pumps iterieren.

Korrektheit des Algorithmus:

Beweis durch Induktion

Sei x_i die Anzahl der berechneten Pumpen mit einem LTC von m_i und O die optimale Lösung für diesen Fall und N_j sind die Pumpstationen. Wir wollen zeigen, dass x_i = O ist.

Bei jedem N_j wissen wir wieviele Pumpen wir bis hier brauchen. Wir kennen die letzte Position der letzten gebauten Pumpe und die letzte Pumpe von N_{j-1}. m_i kann maximal einen Bereich von m_i abdecken deswegen wissen wir, wenn N_j - N_{j-1} > m_i unmöglich ist, oder bei N_j - Position letzte gebaute Pumpe > m_i und bei N_j - Position letzte gebaute Pumpe == m_i eine Pumpe bauen müssen. Somit ist x_{j+1} immer noch optimal.