Temă 1

- la disciplina Structuri de Date -

Maftei Ştefan - Radu
Grupa 314CD
Facultatea de Automatică şi Calculatoare
Universitatea POLITEHNICA din Bucureşti
maftei_stefan_radu@yahoo.com

20 Martie 2016

Cuprins

1 Tema 1 SD - Back to the DNA

1.1 Funcţiile folosite pentru lista circulară

Pentru implementarea problemei am folosit o listă circulară dublu înlănţuită.

Am definit următoarele funcţii pentru manipularea listei:

create() → Alocă memoria şi iniţializează santinela head la NULL, iar length este iniţializat la 0;

pushFront() → Adaugă un element pe prima poziţie în listă, verificând şi cazul în care lista e vidă;

front() → Afiçază primul nod din listă;

popFront() → Şterge primul nod din listă, verificând cazul în care lista e vidă;

pushBack() → Adaugă un element pe ultima poziţie în listă, verificând şi cazul în care lista e vidă;

back() → Afiçază ultimul nod din listă;

popBack() → Şterge ultimul nod din listă, verificând cazul în care lista e vidă;

destroy() → Distruge lista, dezalocând memoria alocată anterior.

Aceste funcţii păstrează proprietatea de listă circulară, iar în cadrul lor se ţine cont de legătura dintre primul şi ultimul nod din listă.

1.2 Implementarea funcţiilor din temă

strToList() → Este adaptor de la un şir de caractere la o listă. Se creează o listvidă cu create(), apoi se parcurge şirul inserându-se caracterul curent cu pushBack() in listă;

listToStr() → Este adaptor de o listă la un şir de caractere. Se alocă memorie pentru şir, folosindu-se câmpul length din listă, apoi cât timp lista are elemente (length > 0) se adauga caracterul în şir, iar elementul din listă este şters cu popFront(). La final se pune '\0';

countNucleotides() → Se creează o listă DNA şi o variabilă n de tipul Nucleotides*. Se obţine lista DNA din şirul de caractere s furnizat cu strToList(), apoi cât timp lista e nevidă (length > 0) se verifică valoarea nodului curent, eliminându-se apoi din listă acel nod cu popFront(). La final se distruge lista şi se returnează n;

computeRNA() → Se creează două liste DNA şi RNA şi o variabilă rna pentru şirul de caractere. Se obţine lista DNA din şirul de caractere s furnizat, apoi cât timp lista DNA nu e vidă se adaugă nodurile în RNA, făcându-se conversia din T în U atunci când e cazul. Se obţine şirul de caractere rna prin listToStr() din lista RNA, se distrug listele şi se returnează rna;

complementDNA() → Se crează două liste DNA şi cDNA şi o variabilă cdna pentru şirul de caractere. Se obţine lista DNA din şirul de caractere furnizat. Cât timp lista nu e vidă, se evalueaza ultimul nod din lista DNA şi se converteşte ca în enunţ, adăgându-use la finalul listei cDNA, nodul fiind eliminat din DNA prin popBack(). Se distrug listele, se obţine şirul de caractere cdna din lista creată cDNA şi se returnează;

countMutations() → Se creează două liste DNA1 şi DNA2 din şirurile de caractere s1 şi s2 furnizate; se iniţializează variabila dif în care calculăm numărul de diferenţe. Ştiind că listele au lungimi egale (secvenţele de ADN sunt egale), cât timp listele sunt nevide, se evaluează nodurile curente şi dacă sunt diferite valorile se incrementează variabila dif; se elimină la fiecare etapă primul nod din fiecare listă cu popFront(). La final se distrug listele şi se returnează dif;

levenshteinDistance() → Se creează două liste DNA1 şi DNA 2 din şirurile de caractere s1 şi s2 furnizate; se parcurge ca la countMutations() până una dintre liste devine vidă, calculând numărul de edit-uri dintre liste. După ce una din ele devine vidă, se mai adaugă la distanţa calculată (numărul de edit-uri) modulul din diferenţa între lungimile rămase ale listelor; se calculează modulul deoarece nu reţinem care e lista care devine vidă. Se distrug listele şi se returnează distanţa calculată.