Structuri de date și algoritmi

Universitatea Politehnica Timișoara
P-ța Victoriei nr. 2
RO 300006 - Timișoara
Tel: +4 0256 403000
Fax: +4 0256 403021
rector@rectorat.upt.ro
www.upt.ro

Domeniul de studii: Informatică/ Specializarea: Informatică

SDA – Cursul 9

Ș.l. dr.ing. Adriana ALBU
adriana.albu@upt.ro
http://www.aut.upt.ro/~adrianaa

Universitatea
Politehnica
Timişoara

P-ţa Victoriei nr. 2
RO 300006 - Timişoara
Tel: +4 0256 403000
Fax: +4 0256 403021
rector@rectorat.upt.ro
www.upt.ro

4. Structura de date șir

4.1 TDA șir

- MM: secvență finită de caractere $c_1, c_2, \ldots, c_n$
  - n -> lungimea secvenței; n=0 => șir vid
- Notații:
  - s, sub, u – șiruri
  - c – valoare de tip caracter
  - b – boolean
  - poz, lung – întregi pozitivi
- Operatorii: implementați în termenii limbajului de programare (primitivi) sau dezvoltați de programator
- Operatori:
  - CreeazaSirVid(s)
  - InsereazaSir(sub,s,poz)
  - SirVid(s)->b
  - FurnizeazaCar(s,poz)->c
  - AdaugaCar(s,c)
  - LungSir (s)->lung
  - Pozitie(sub,s)->poz
  - StergeToateSubsir(sub,s)
  - Concat(u,s)
  - StergeSir(s,poz,lung)
  - CopiazaSubsir(u,s,poz,lung)

4.2 Implementarea TDA sir
4.2.1 Implementarea cu tablouri

- articol format dintr-un întreg (lungimea șirului) și un tablou de caractere

```
#define LungMax …
typedef int TipLungime; /*0… LungMax*/
typedef int TipIndice;  /*0… LungMax-1*/
typedef char tab[LungMax];
typedef struct{
     TipLungime lungime; /*pentru eficiență*/
     tab sir;
}TipSir;
TipSir s;
```

- Ex.:
  - Creeaza_Sir_Vid(TipSir s); Lung_Sir (TipSir s); Adauga_Car(TipSir s, char c); → O(1)
  - Copiaza_Subsir(TipSir u, TipSir s, TipIndice poz, TipLungime lung); → O(n)
- Implementare simplă, operatori performanți, dar cu o utilizare ineficientă a spațiului de memorie

4.2.1 Implementarea cu tablouri

➢Exemple din C, biblioteca string.h

```
char str[50];

strcpy(str,"Ana are mere");
printf("%s",str); // Ana are mere

printf("%c",str[4]); // a
str[8]=toupper(str[8]); // => Ana are Mere

strcat(str,"!"); // => Ana are Mere!

printf("%s",strchr(str,'n')); // na are Mere!
printf("%s",strstr(str,"are")); //are Mere!
```

➢ineficient

```
for(i=0;i<strlen(str);i++)
      //…
```

➢mai bine

```
int n=strlen(str);
for(i=0;i<n;i++)
      //…
```

4.2.2 Implementarea tipului șir prin tabele

- Toate șirurile utilizate la un moment dat se păstrează într-un singur tablou numit *Heap*
- Se asociază o tabelă auxiliară (tablou auxiliar) care păstrează evidența șirurilor, numita tabelă de șiruri
  - o locație în tabela de șiruri referă un șir printr-un articol compus din 2 elemente:
    - lungimea șirului curent
    - un indicator în *Heap* care precizează poziția (începutul) șirului

4.2.2 Implementarea tipului șir prin tabele

```
#define LungimeHeap …
#define LungimeTabela …
typedef struct{
     int lungime;/*0…LungimeHeap*/
     int indicator;/*0…LungimeHeap-1*/
}ElemTablou;
typedef int TipSir;
typedef ElemTablou tab1[LungimeTabela];
typedef char tab2[LungimeHeap];
tab1 TabelaDeSiruri;
tab2 Heap;
int Disponibil;
int NumarDeSiruri;
```

- Lungimea limitată a șirurilor complică operatorii de tip adăugare caracter, ștergere subșir etc.
  - → noi instanțe ale șirurilor implicate, la sfârșitul tabloului *Heap*
- Copierea unui șir sursă într-un șir destinație se obține poziționând câmpul indicator al destinației spre indicatorul șirului sursa în *Heap*
  - → economie de memorie

4.2.3 Implementarea șirurilor cu pointeri

- Se bazează pe reprezentarea șirurilor drept o colecție de celule înlănțuite, fiecare celulă conținând unul sau mai multe caractere
- Accesul la caracterele șirului se face doar secvențial

```
typedef struct cel{
     char ch;
     struct cel *urm;
}Celula;
typedef Celula *PointerCelula;
typedef struct{
     int lungime;
     PointerCelula cap, coada;
}TipSir;
TipSir s;
```

- Se utilizează în special când lungimea șirurilor este imprevizibilă

4.3 Tehnici de căutare / 4.3.1 Căutare tabelară

- Pentru a stabili coincidența a două șiruri e necesar să se stabilească coincidența tuturor caracterelor corespunzătoare din șirurile comparate

```
#define M …
typedef char sir[M];
sir x,y;
```

$x==y \longleftrightarrow A_j: 0 <= j < M : x_j == y_j$

- Lungimea unui șir poate fi precizată:
  - explicit (implementările anterioare)
  - implicit, prin precizarea unui caracter fanion care să marcheze terminarea șirului
- **Căutarea tabelară** parcurge un tablou de șiruri, pentru fiecare intrare în tablou verificând prezența șirului căutat `x` prin aplicarea tehnicii de căutare liniară
  - în cazul unui terminator de șir coincidența apare dacă ultimul caracter identic este terminatorul de sir
- Structuri imbricate

4.3 Tehnici de căutare în șiruri / 4.3.2 Căutarea directă

- Are drept scop stabilirea poziției primei apariții a șirului `p` în șirul `s`
  - rezultatul căutării este indicele `i` al primei apariții

```
char s[N];
char p[M]; /* M<<N*/
```

- Precizarea coincidenței se face cu P(i,j):

`P(i,j)` A_k: `0 <= k < j :` $s_{i+k} == p_k$

- În cazul în care P(i,j) este adevărat, avem o coincidență de `j` caractere începând cu poziția `i` în șirul `s`
  - pentru a avea o coincidență pe lungimea M este necesar să fie adevărat P(i,M)
- Șirul `p` este deplasat paralel cu șirul `s` până la găsirea lui `p` sau până când numărul caracterelor netestate din `s` este mai mic decât dimensiunea lui `p`

4.3 Tehnici de căutare în șiruri / 4.3.2 Căutarea directă

- Căutarea directă lucrează destul de repede dacă se presupune că nepotrivirea dintre `s` și `p` apare după cel mult câteva comparații
  - → O(N)

s: ABABCDE ABABCDE ABABCDE

p: ABCD ABCD ABCD

- Cel mai defavorabil caz este când nepotrivirea dintre `s` și `p` apare totdeauna pe ultimul caracter din `p` → O(N*M)
  - s: AAAAAAAAAAAAAA.....AAAAAA
  - p: AAAAAAAAAB
- Eficiență scăzută: modelul `p` avansează cu o singură poziție după o nepotrivire

4.3 Tehnici de căutare în șiruri / 4.3.3 Optimizări

- Optimizările presupun preprocesare
  - Căutarea Boyer-Moore
    - se bazează pe ideea de a realiza comparația dintre modelul p și șirul s, începând cu ultimul caracter al modelului
  - Căutarea Knuth-Morris-Pratt
    - ia în considerare sub-modele ale modelului căutat
    - preprocesarea – identificarea celor mai lungi secvențe de caractere din model care sunt prefix al modelului
    - la medie, performanțe O(N+M)
    - https://www.youtube.com/watch?v=cH-5KcgUcOE

Universitatea
Politehnica
Timişoara

P-ţa Victoriei nr. 2
RO 300006 - Timişoara
Tel: +4 0256 403000
Fax: +4 0256 403021
rector@rectorat.upt.ro
www.upt.ro

Vă mulțumesc!