Претраживање стрингова

др Ђура Паунић
Институт за математику, ПМФ Нови Сад

Основни проблем који се решава у претраживању стрингова је да се испита *да ли се у датом стрингу налази други стринг као подстринг*. Овај проблем се у програмирању јавља довољно често, на пример када желимо да пронађемо низ знакова у фајлу који садржи ASCII текст. Да би се описани алгоритми могли лако пренети у друге програмске језике за реализацију стринга нећемо користити постојећи тип `string` у Pascalu , него ће се он дефинисати као посебан кориснички тип стринга `tstring`. Претпоставимо да је стринг дефинисан на најједноставнији начин као слог у коме се знаци налазе у низу, у пољу `z`, а дужина стринга (број знакова) је број у пољу `d`.

```
const
  MaxString = 500;

type
  znaci : array[1 .. MaxString] of char;
  tstring = record
              z : znaci;
              d : integer
            end;
```

За објашњења ће се у стрингу s поједини знаци означавати индексима,

$$s = s_1 s_2 s_3 \ldots s_k,$$

при чему је k дужина стринга. Нека су $p = p_1 p_2 \ldots p_m$ и $s = s_1 s_2 s_3 \ldots s_k$ стрингови и нека је $m \leq k$. Стринг p се назива подстринг стринга s ако постоји индекс n $1 \leq n \leq k - m$ такав да је $s_{n-1+i} = p_i$ за $i = 1, \ldots, m$. Тада кажемо да се подстринг p појављује на n-том месту у стрингу s. Подстринг може да постоји у стрингу који се претражује на више различитих места. Без ограничења општости се може ограничити на налажење првог појављивања подстринга у стрингу. При претраживању се претпоставља да је резултат претраживања позиција првог знака подречи у стрингу која је једнака подстрингу или 0 ако у стрингу таква подреч не постоји.

Приликом објашњавања алгоритама претраживања ће се стринг који се претражује увек означавати са s, а са p стринг који се тражи у стрингу s као подстринг.

Најједноставнији начин да се испита да ли је стринг p подстринг стринга s је да се знаци у стрингу p редом упоређује са подречима у стрингу s. Дакле, применити следећи поступак. Поставити почетак стринга који се тражи p поравнато са почетком стринга s, у коме се p тражи, тако да им се први знаци поклопе. Редом упоређивати одговарајуће знаке у њима, први са првим, други са другим итд. Ако су сви знаци у p једнаки одговарајућим знацима у стрингу s подреч је пронађена. Ако нису, померити p тако да први знак p буде поравнат са другим знаком у стрингу s и опет редом упоређивати знаке.

Сада је једноставно написати функцију **BFSearch** (BruteForceSearch) која реализује овај поступак.

Ако је дужина стринга s мања од дужине стринга p тада s очигледно не садржи p. У промењивој **mesto** је позиција почетка подречи у стрингу s која се упоређује са p, а у промењивој **test** је позиција знака у p који се упоређује са знаком у стрингу s. Погодно је да **mesto** има предност за један мање од праве позиције знака, јер је тада **mesto + test** позиција знака у s који се упоређује са **p.z[test]**. Последња почетна позиција на којој треба тражити стринг p у стрингу s је **s.d - p.d + 1** тако да када промењива **mesto** добије предност **kraj = s.d - p.d + 1** *тада се p не може више налазити у s* (**mesto** *је за један мање од* правог места).

```
function BFSearch(s, p : tstring) : integer;

var
  test, mesto, kraj : integer;

begin
  if s.d < p.d then
    BFSearch := 0
  else begin
    mesto := 0;
    test := 1;
    kraj := s.d - p.d + 1;
    repeat
      if p.z[test] = s.z[test + mesto] then
        test := test + 1
      else begin
        mesto := mesto + 1;
        test := 1
      end
    until (test > p.d) or (mesto = kraj);
    if test > p.d then
      BFSearch := mesto + 1
    else
      BFSearch := 0
  end
end; (* BFSearch *)
```

Несумњиво је да се може направити бољи алгоритам, јер се у овом алгоритму приликом наиласка на знаке који нису међусобно једнаки све заборавља и почиње упоређивање од почетка, уместо да се искористи информација добијена при делимичном слагању стрингова. Ако један исти стринг p треба да се упоређује више пута са подречима у стрингу s тада се исплати уложити нешто времена у препроцесирање стринга p да се добије додатна информација која може да се искористи за ефикасније претраживање од секвенцијалног упоређивања знакова.

Постоји низ алгоритама који су бржи од претраживања грубом силом од којих су најпознатији Кнут-Морис-Пратов алгоритам или Бојер-Муров алгоритам из

1977. године и разна њихова побољшања, али су они компликовани. Изложимо један ефикасан алгоритам који је и једноставан.

Д. М. Санди (Sunday) је 1990. предложио алгоритам за претраживање стринга који је назвао Брзо тражење (енгл. QuickSearch) . Основна идеја у алгоритму брзог тражења је врло проста. Стринг p, који се тражи у стрингу s, упоређује се са подречима у стрингу s као и у претраживање грубом силом. Ако дође до неслагања два знака тада треба померити стринг p бар за једно место удесно. Дакле, у следећем покушају треба да се упоређује знак који је у стрингу s иза последњег знака стринга p са неким знаком у стрингу p. Назовимо знак који се налази у стрингу s иза последњег знака стринга p *пробни знак*. Ако се пробни знак не налази у p тада се p може померити тако да се почетак p поравна са знаком иза пробног знака, дакле за $p.d + 1$. Уколико се пробни знак налази у стрингу p тада се p помера за минималну вредност за коју се пробни знак поклапа са знаком у p (тако да се крајње десно појављивање пробног знака у p поклопи са пробним знаком у s).

Претраживање Брзим тражењем се изводи на исти начин као и у претраживању грубом силом осим што се у случају неслагања не помера за 1 него за износ који зависи од позиције пробног знака и који треба да се претходно израчуна. Најједноставније је тај износ памтити у низу **shift**. Дакле, треба за сваки знак у азбуци израчунати последњу позицију тог знака у стрингу p. Нека се ове позиције налазе у низу **znak** индексираном редним бројевима знакова. Вредности овог низа се врло лако израчунавају.

Ако се води рачуна да стринг p не може да се налази у стрингу s када је дужина s мања од дужине p и да је тражење просто када се стринг p састоји од само једног знака добија се функција:

```
function QuickSearch(s, p : tstring) : integer;

var
  shift : array[32 .. 256] of integer;
  ch : char;
  i, offset, test, mesto, kraj : CARDINAL;

begin
  if s.d < p.d then
    RETURN 0
  else begin
    if p.d = 1 then begin
      mesto := 1;
      ch := p.z[1];
      while (mesto <= s.d) and (s.z[mesto] <> ch) do
        mesto := mesto + 1;
      if mesto > s.d then
        QuickSearch := 0
      else
        QuickSearch := mesto
    end
    else begin
```

```
      offset := p.d + 1; (* pozicija probnog znaka *)
      for i := 32 to 256 do
        shift[i] := offset;
      for i := 1 to p.d do
        shift[ord(p.z[i])] := offset - i;
      mesto := 0;
      test := 1;
      kraj := s.d - p.d + 1;
      repeat
        if p.z[test] = s.z[test + mesto] then
          test := test + 1
        else begin
          mesto := mesto + shift[ord(s.z[mesto + offset])];
          test := 1
        end
      until (test = p.d) or (mesto >= kraj);
      if test = p.d then
        QuickSearch := mesto + 1
      else
        QuickSearch := 0
    end
  end
end;  (* QuickSearch *)
```

Статијата прв пат е објавена во списанието Тангента на ДМ на Србија во 1999/00 година