Struktura Trie (čita se kao engl. *try*)

- naziv dolazi iz *retrieval* (ponovni dohvat, povrat)
  - Fredkin, Edward (Sept. 1960), "Trie Memory", *Communications of the ACM* **3**(9), pp. 490-499
- Trie - stablo za čije se pretraživanje koriste samo dijelovi ključeva pohranjenih podataka
- slična prefiksnom (*prefixed*) B^+-stablu, ali nema komplikacija s određivanjem optimalnih prefiksa
  - ponovimo: B^+-stabla se uvode radi bržeg slijednog pristupa podatcima; unutarnji čvorovi ne sadrže informacije, već ključeve i služe samo za brzo pretraživanje (*index set*); listovi su međusobno povezani u niz (listu, engl. *sqeuence*) i čine tzv. *sequence set*; dakle, B^+-stablo je indeks u obliku B-stabla plus lista svih podataka u stablu pa mu odatle i naziv;
    prefiksno B^+-stablo razvija tu ideju dalje pa indeksi nisu cijeli ključevi, nego najmanji dijelovi ključa (prefiksi) dovoljni za razlikovanje dva susjedna podatka
- dvije vrste čvorova: unutarnji - samo kôdovi (svi dijelovi svih ključeva) i pokazivači, listovi - podatci

Prednost pred stablima (u nekim primjenama)

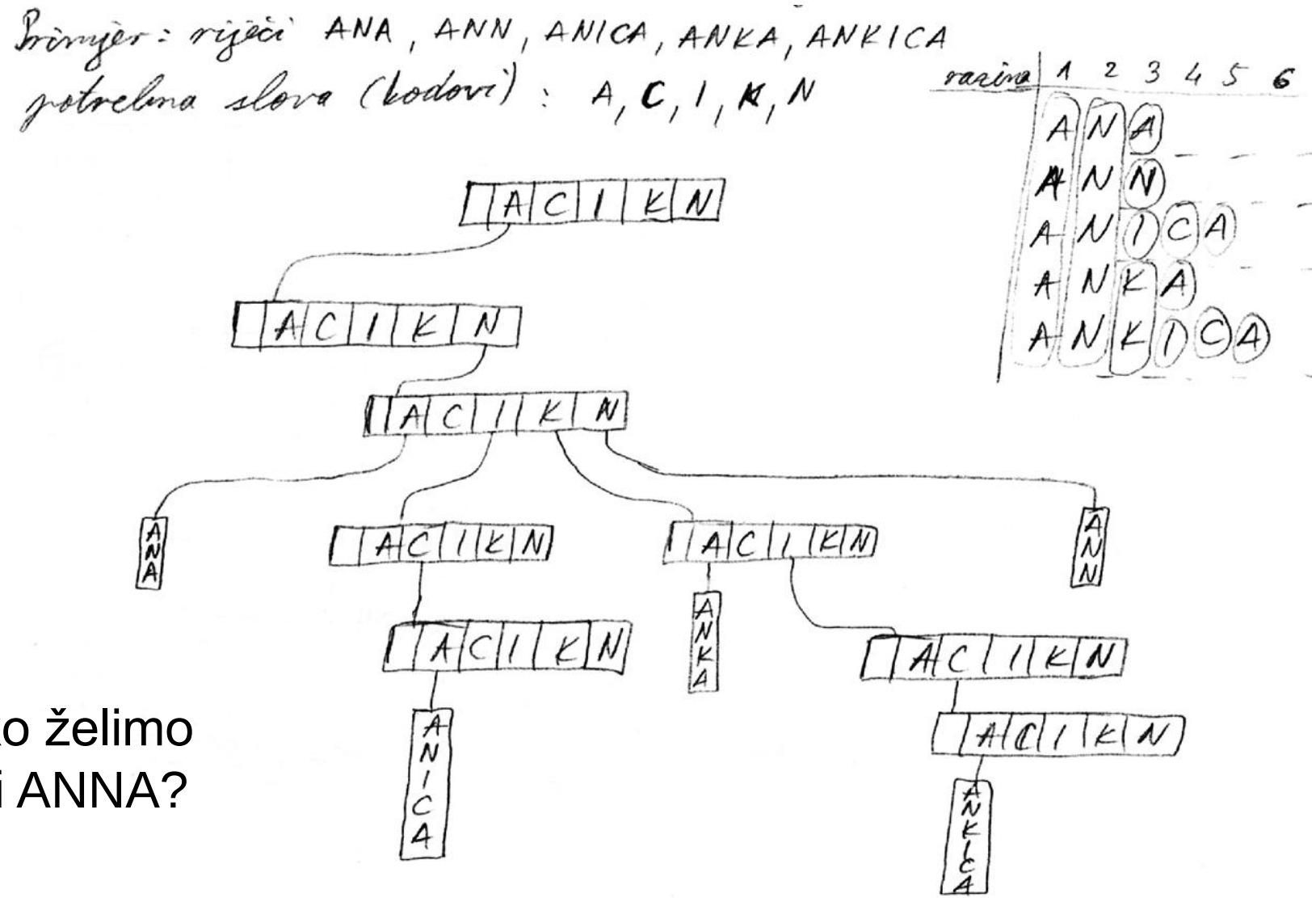
- ovisno o svojstvima podataka (npr. *spell checker* koji mora brzo pronaći relativno kratak string u skupu od više desetaka tisuća riječi) bolja od stabla jer jedan čvor ima više od dva podstabla pa je niža i pristup podatcima je brži (visina joj je jednaka najdužoj riječi; u HR i EN riječi su relativno kratke)
- konačni oblik Trie ne ovisi o redoslijedu upisa podataka
- svaka pojedinačna usporedba je brža jer se ne provjerava cijeli ključ, nego samo jedan njegov element
- budući da su svi podatci u listovima i nisu međusobno povezani, brisanje je vrlo jednostavno

Nedostatak u usporedbi sa stablima

- zbog dvije vrste čvorova, dodavanje elemenata je kompliciranije nego u stablu
  (ne jako komplicirano - vidi Drozdek...)

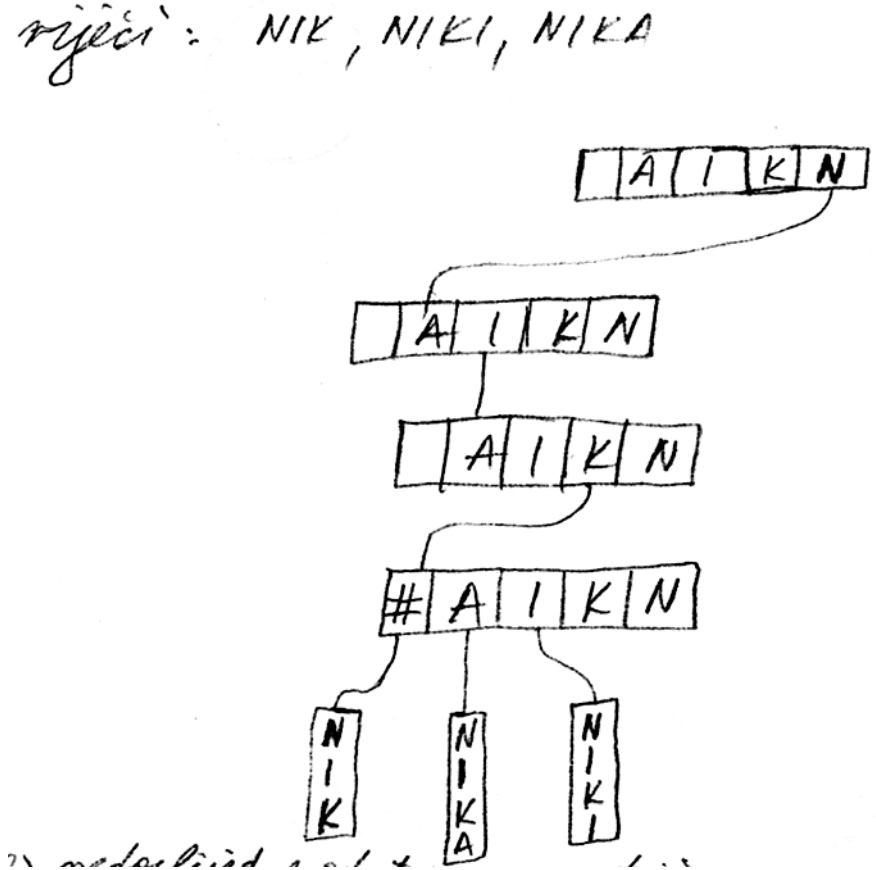
Pretraživanje: iz čvora na k-toj razini ide se u čvor koji pokazuje pokazivač uz kôd jednak k-tom dijelu ključa koji se traži.

- Ako je pokazivač NULL, traženog podatka nema (mjesto za ubacivanje).



Što ako želimo upisati ANNA?

Rješenje: svakom čvoru dodati poseban kôd (u primjeru #) koji sigurno nije dio nijednog ključa i koji će značiti kraj riječi. Kada se prilikom pretraživanja dođe do posljednjeg elementa ključa (posljednjeg slova), pokazivač nas mora poslati u # dio čvora na nižoj razini, gdje je smješten prefiks, tj. riječ koja završava posljednjim slovom ključa.



Da listovi ostanu listovi (druga vrsta čvorova), ako nema prefiksa, pokazivač mora biti usmjeren izravno u podatak, a ne niži indeks-čvor. Npr., prvo se upiše NIK, a onda NIKI. NIK treba biti normalno smješten, tj. K-pokazivač na trećoj razini treba pokazivati podatak NIK. Kada dođe NIKI, K-pokazivač na trećoj razini treba preusmjeriti na novi indeks-čvor. U novom čvoru, “I” treba usmjeriti na NIKI, a # na NIK.

Praćenje unatrag (*Backtracking*)

- algoritam koji "pamti" slijed koraka (prethodne pokušaje) tijekom rješavanja problema pa se nakon neuspješnog pokušaja može vratiti na polazno stanje i nastaviti samo s još neisprobanim mogućnostima
- za probleme čije rješavanje iz koraka u korak nudi više izbora za sljedeći korak

Tipičan primjer:
problem kraljica
(ponoviti ASP!)

- *backtracking* se ostvaruje `for` petljom, tj. povratkom u nju nakon nuspješnog pokušaja

```
Queen (row)
for every position col in the same row;
   if col is not attacked by already placed queens
   {   place the queen in position col;
       if row < 8
              Queen(row+1);
       else
              return success;
       remove the queen from position col;  }
return failure;
```

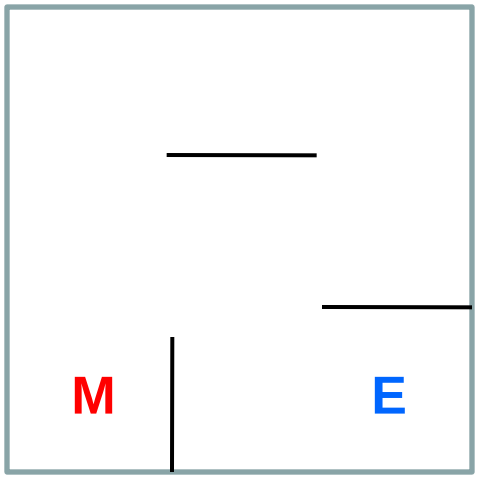
Glavna odlika i prednost *backtrackinga* je izbjegavanje moguće velikog broja (neisprobanih) mogućnosti ako se ustanovi da one sigurno ne vode ka rješenju.

- kraljice: ako se kraljica uopće ne može smjestiti u npr. peti redak (svugdje ju napadaju već postavljene), redci 6, 7 i 8 se ni ne ispituju, nego se traži novo mjesto (stupac) za kraljicu u četvrtom retku

Backtracking će nas sigurno dovesti do rješenja ako ono postoji, ali pomoći u izbjegavanju jalovih pokušaja može samo kada se konačni neuspjeh može predvidjeti prije iscrpljivanja svih mogućnosti koje pruža posljednja donesena odluka.

- kraljice: nakon neuspješnog pokušaja smještanja 5. kraljice, odmah smo znali da je smještaj 4. kraljice bio pogrešan (takav da sigurno ne možemo naći rješenje)

U npr. problemu izlaska iz labirinta, *backtracking* ne smanjuje broj mogućnosti koje moramo isprobati.



1	1	1	1	1	1	1
1	0	0	0	0	0	1
1	0	0	1	0	0	1
1	0	0	0	0	0	1
1	0	0	0	0	1	1
1	M	1	0	0	E	1
1	1	1	1	1	1	1

P = prolaz
Pmax = 3

Z = zid
Zmax = Pmax – 1 = 2

D = dim = Pmax + Zmax = 5

```
exitMaze (cell)                    //rekurzivno - tail rekurzija
if cell = exitCell
   return success;
if exitMaze(up) == success or exitMaze(down) == success
    or exitMaze(left) == success or exitMaze(right) == success
       return success;
return failure;
```

```
exitMaze ()                        //nerekurzivno
initialize stack, entryCell, exitCell, currCell = entryCell
while currCell is not exitCell
{  mark currCell cell as visited;
   push onto the stack the unvisited neighbours of currCell;
   if stack is empty
       return no way out;
   else
       currCell = a cell from the stack;    }
return success;
```

Maze

Odabrani algoritmi nad grafovima

Osnovne definicije

Graf (engl. *graph*) - unija nepraznog skupa vrhova V (engl. *vertex*) i moguće praznog skupa bridova E (engl. *edge*); G=(V,E).

- graf je zapravo stablo (šuma, engl. *forest*) u kojem nema hijerarhije među čvorovima

Brid je svaki dvočlani podskup skupa V.

- poveznica dva vrha

Red (engl. *order*) grafa je broj vrhova u njemu.

- označava se s |V|

Veličina (engl. *size*) grafa je broj bridova u njemu.

- označava se s |E|
- brid između vrhova v_i i v_j označavamo s edge(v_i, v_j) ili kraće samo v_i v_j

Povratna petlja - brid koji počinje i završava u istom vrhu.

Supanj (engl. *degree*) vrha - broj bridova koji su prileže ći (engl. *incident*) tom vrhu (spojeni s njim).

- stupanj vrha v označavat ćemo s deg(v)
- povratne petlje se u deg(v) ubrajaju dva puta

Razlikujemo:

1. neusmjereni (engl. *undirected*) graf - za svaki brid $v_i v_j$ vrijedi $v_i v_j = v_j v_i$
   - vrhovi se smatraju susjednima (engl. *adjacent*) ako je brid $v_i v_j$ u E
   - takav se brid naziva prileže ćim (engl. *incident*) vrhovima v_i i v_j
2. usmjereni (engl. *directed*) graf - za brid $v_i v_j$ ne mora vrijediti $v_i v_j = v_j v_i$
   - vrh v_j se smatra susjedom vrhu v_i ako postoji brid brid $v_i v_j$ u E (ako se iz v_i može izravno u v_j)
   - vrh v_i se smatra susjedom vrhu v_j ako postoji brid brid $v_j v_i$ u E
   - brid $v_i v_j$ se naziva izlaznim bridom (engl. *outedge*) vrha v_i i ulaznim bridom (engl. *inedge*) vrha v_j

Daljnje definicije i podjele:

Jednostavni graf (engl. *simple graph*) - graf koji između svaka dva vrha ima najviše jedan brid i u kojem nema povratnih petlji (slike. a-d).

- taj se pojam primarno odnosi na neusmjerene grafove

Multigraf (engl. *multigraph*) - graf koji može imati više od jednog brida između dva vrha (slika e).

Pseudograf (engl. *pseudograph*) - multigraf u kojem mogu postojati povratne petlje (slika f).

Staza ili obilazak (engl. *path*) od vrha v_1 do vrha v_n je niz bridova v_1v_2, v_2v_3, …, $v_{n-1}v_n$.

- kraće se označava $v_1,v_2,...,v_{n-1}$ ili samo $v_1v_2...v_n$

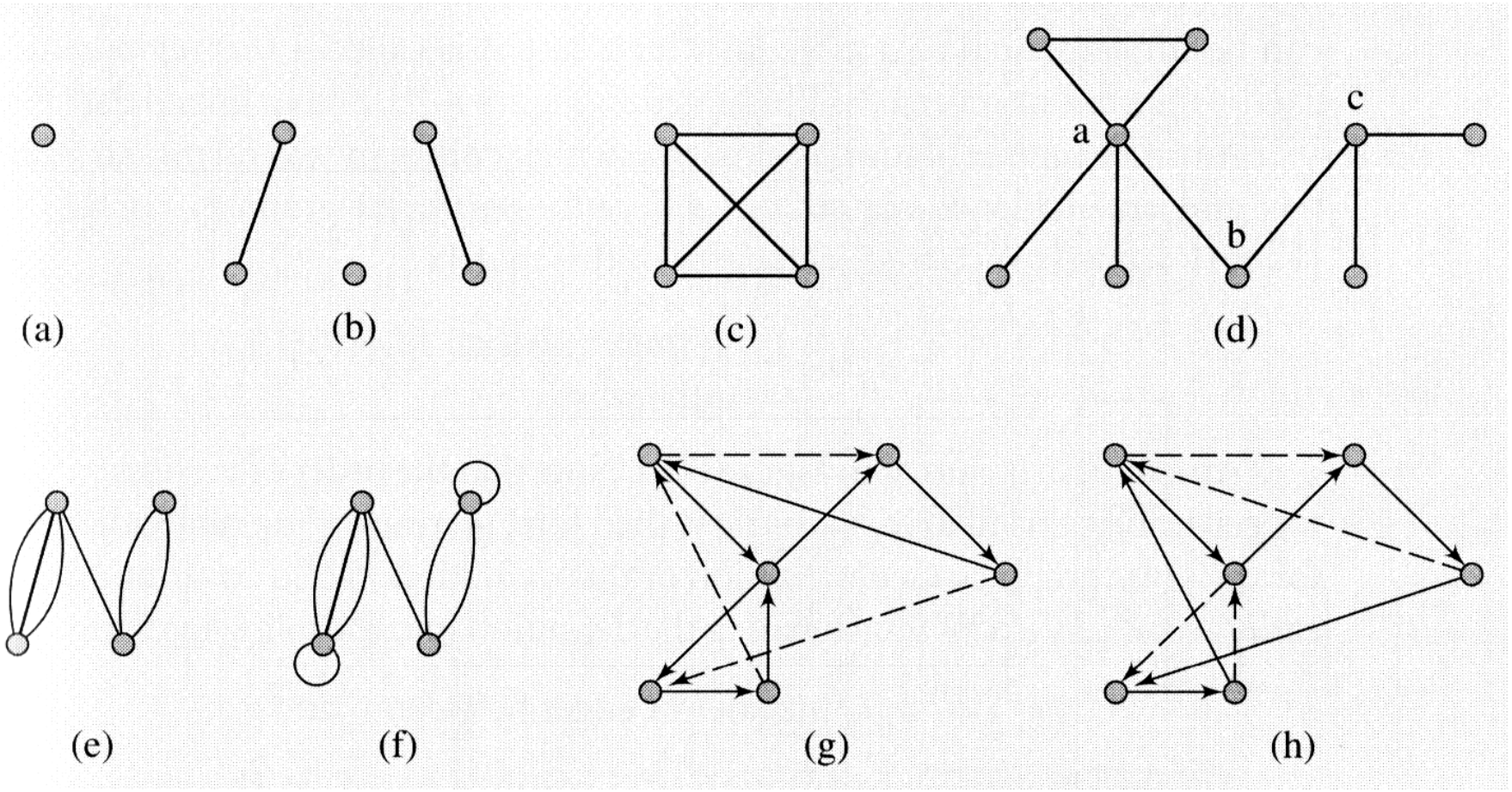
Put ili putanja - staza u kojoj su svi bridovi različiti (znači svakim bridom se prolazi samo jednom, ali vrhovi se mogu ponavljati).

Petlja ili krug (engl. *circuit*) - put u kojem je $v_1 = v_n$ (slika g).

Ciklus (engl. *cycle*) - petlja (*circiut*) u kojoj su svi vrhovi, osim v_1 i v_n, različiti (slika h).

Primjeri uz definicije pojmova iz teorije grafova:

slike a-d: jednostavni grafovi, e - multigraf, f - pseudograf, g - petlja, h - ciklus



Potpuni (engl. *complete*) graf - graf u kojem je između svaka dva vrha točno jedan brid.

- odnosi se samo na neusmjerene grafove
- potpuni graf n-tog reda označava se s Kn
- broj bridova u potpunom grafu = broj dvočlanih podskupa skupa vrhova V

$$|E| = \binom{|V|}{2} = \frac{|V|!}{(|V|-2)!\cdot 2!} = \frac{|V|(|V|-1)}{2} = \mathrm{O}(|V|^2)$$

Podgraf (engl. *subgraph*) - skup G' = (V',E') koji je podskup grafa G = (V,E) takav da svaki brid iz E' pripada i E.

Prikaz (pohrana) grafa na računalu

1. Lista susjedstva (engl. *adjacency list*) - svi vrhovi susjedni nekom vrhu
   a) u obliku tablice (engl. *star representation*) - slika b
   b) kao dvodimenzionalno povezana lista - slika c
2. Matrični zapis
   a) Matrica susjedstva (engl. *adjacency matrix*) - [vrhovi x vrhovi]: slika d
   b) Matrica povezanosti (engl. incidence matrix) - [vrhovi x bridovi]: slika e

Odabir ovisi o situaciji:

a) pristup svim susjedima nekog vrha - bolje liste jer je potrebno deg(v) koraka, naspram |V| za matrice
b) pojedinačne intervencije (dodati ili ukloniti vrh) - bolje matrice; brži pristup i O(1) održavanje

Prikazi grafa:

b,c - lista susjedstva

d - matrica susjedstva

e - matrica povezanosti