

MOHÓ módszer

8. ELŐADÁS

Sapientia-EMTE

2020-21

„Mohó kódok”

- 1951-ben, az MIT-n, David Huffman és osztálytársai választhattak a vizsga és egy tudományos dolgozat írása között, „leghatékonyabb bináris kód” témakörben.
- Huffman már azon volt, hogy nekilát tanulni a vizsgára, amikor hirtelen az az ötlete támadt, hogy használjon *gyakoriságon alapuló bináris fát* a kódoláshoz, amelyről könnyűszerrel bizonyítani is tudta, hogy a lehető leghatékonyabb módszer.

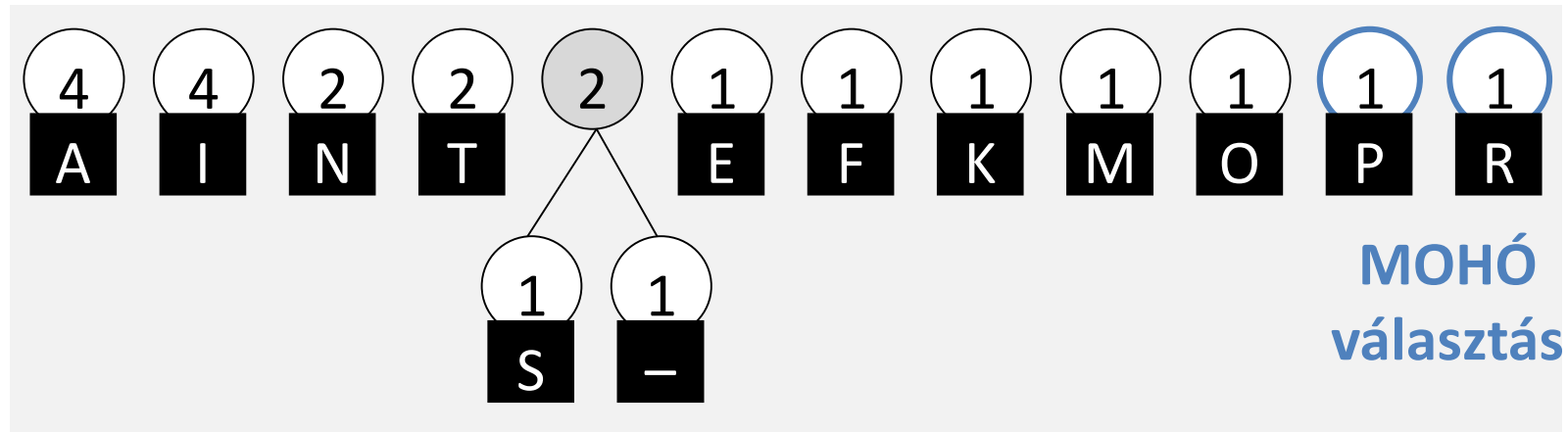
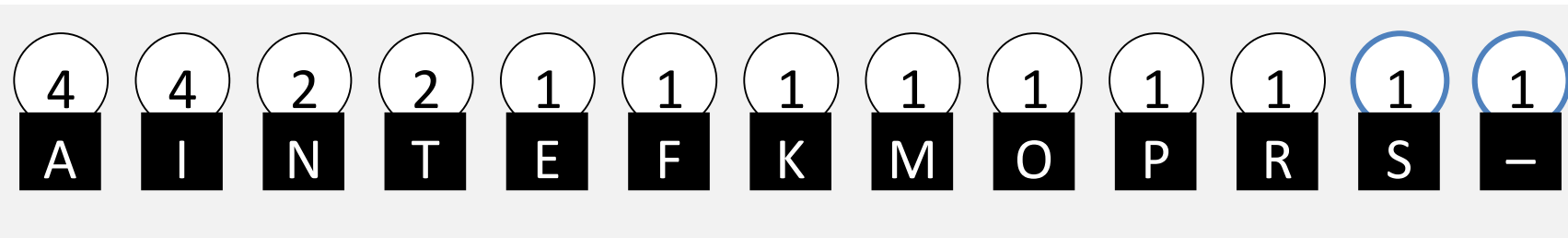


Huffman – fa (0)

- Huffman egy bináris fát épített a gyakorisági tábla alapján:
 - A szöveg karakterei a fa leveleibe kerülnek;
 - A megfelelő kódok a gyökér-levél utak bináris ábrázolásai lesznek.
- Huffman a fát levelektől gyökér fele irányba építette fel:
 - Kezdetben minden karakter csomópontja különálló egy pontú fának tekintendő;
 - Minden lépésben összevonjuk a két legkisebb gyakoriság értékű gyökérrel rendelkező részfat oly módon, hogy létrehozunk részükre egy közös apa-csomópontot;
 - Az új apa-csomópont gyakoriságértéke, a fiú-részfák gyökerei gyakoriságértékeinek összege lesz.

Huffman – fa (1)

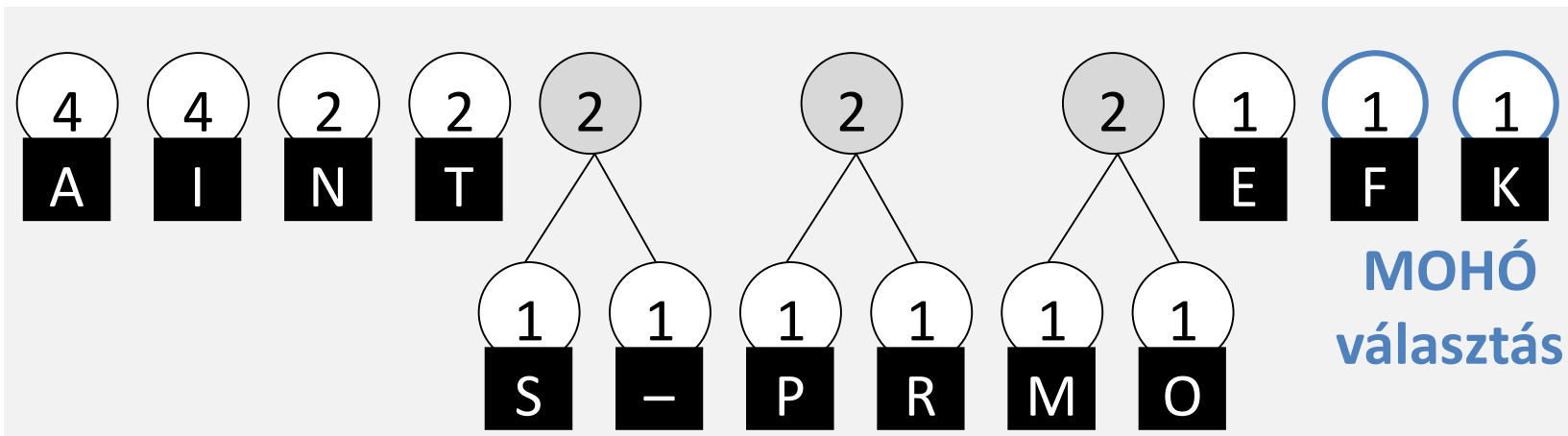
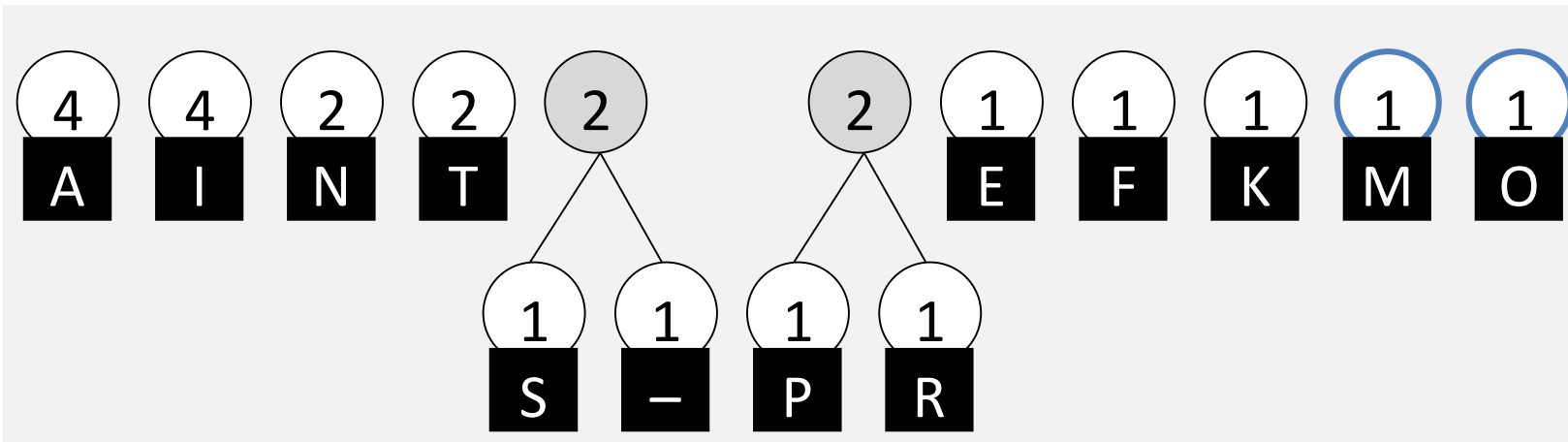
MOHÓ
választás



MOHÓ
választás

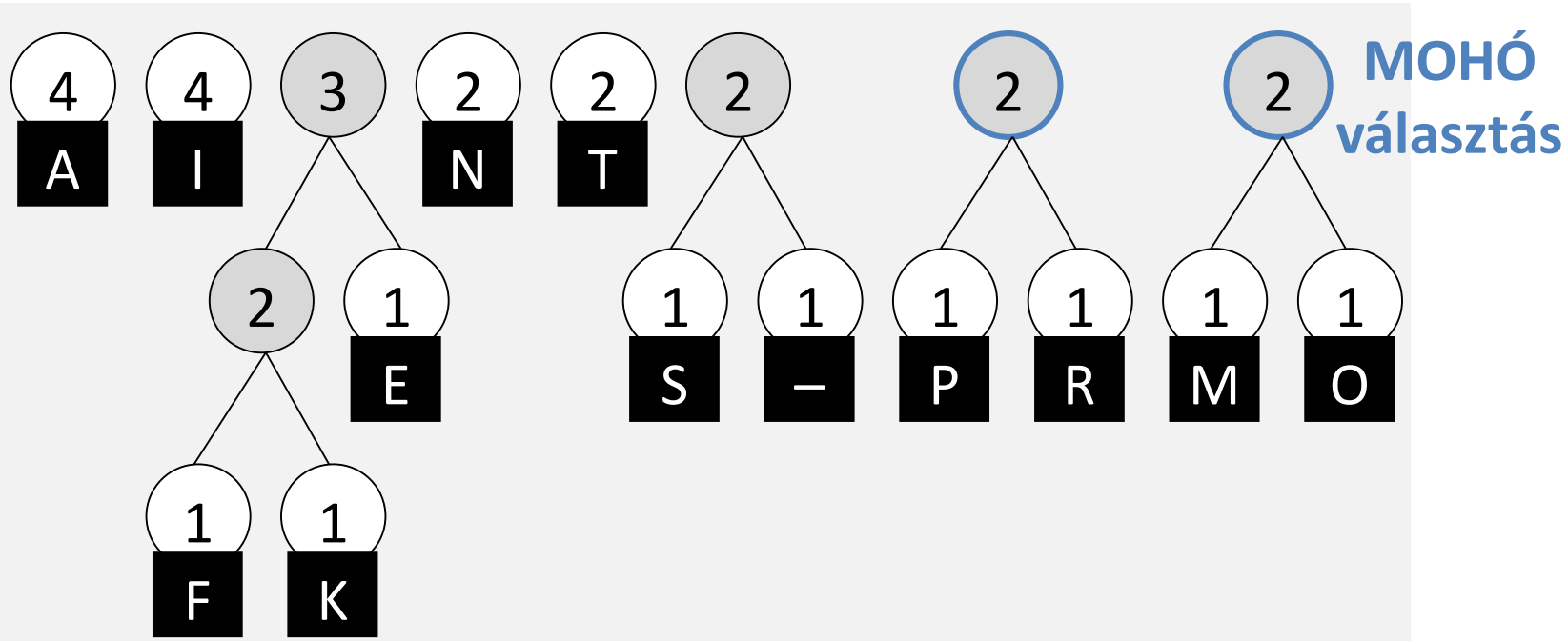
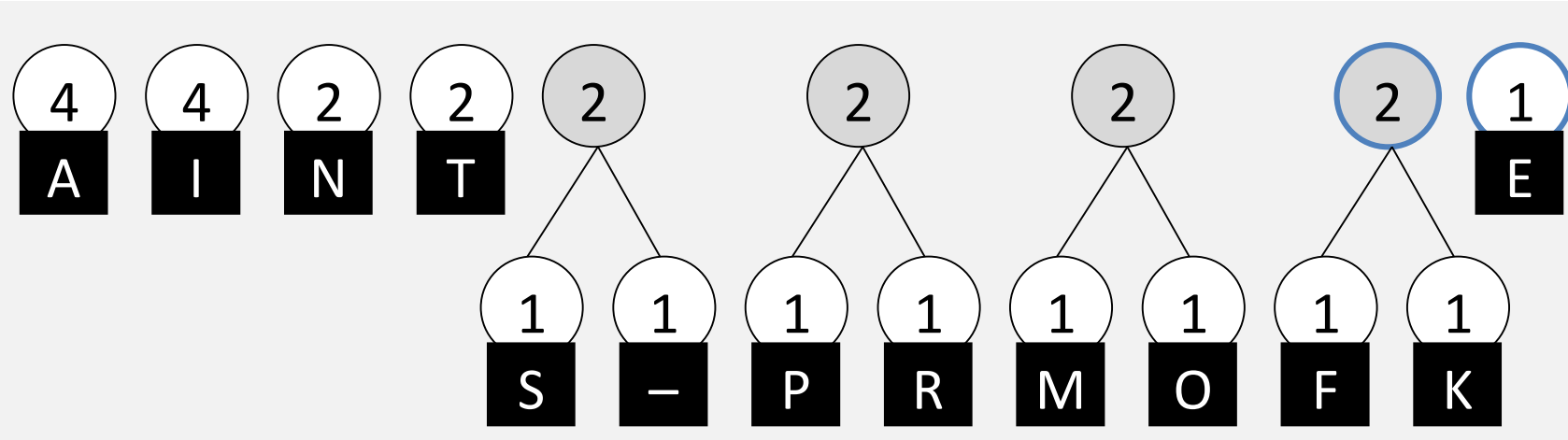
Huffman – fa (2)

MOHÓ
választás



Huffman – fa (3)

MOHÓ választás



[illegible]

A (4) : 000

I (4) : 001

N (2) : 011

F (1) : 01000

$$R(1) : 1001$$

K (1) : 01001

$$-(1) : \begin{array}{cccc} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{array}$$

Huffman – kódolás

Naiv kódolás:

- 13 féle karakter
- 4-bit hosszú karakter-kódok
- $21 \times 4 = 84$ hosszú kódolt szöveg

74 bit

SAPIENTIA-INFORMATIKA

S (1) : 1110

A (4) : 000

P (1) : 1000

I (4) : 001

E (1) : 0101

N (2) : 011

T (1) : 110

F (1) : 01000

O (1) : 1011

R (1) : 1001

M (1) : 1010

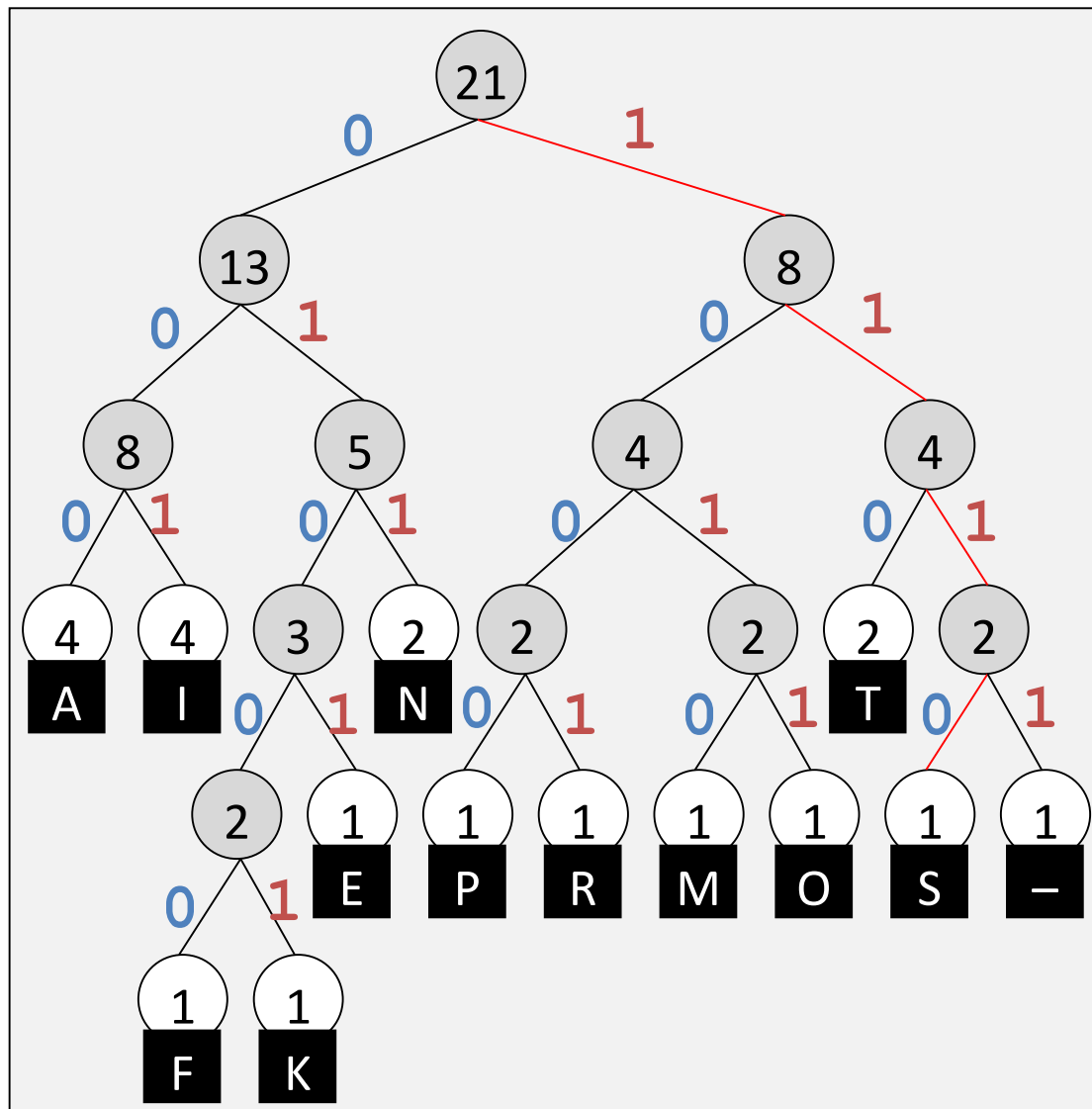
K (1) : 01001

- (1) : 1111

11100001000001010101111000100011110010110100010111001101000011000101001000

Dekódolás

Mivel a kódok
kizárólag levelekbe
vezető utak, ezért
egyik se prefix-je a
másiknak



S

11100001000001010101111000100011110010110100010111001101000011000101001000

HUFFMAN: kódolás/dekódolás

Implementációs ötletek (1)

```
int stat[256] = {0}, n = 0;
char c;
while ( fscanf(fin, "%c", &c) != EOF ) {
    ++stat[c];
    if( stat[c] == 1 ) { ++n; }
}
stat[27] = 1; ++n;
```

stat	0	0		1		2		1		2		1		1		1		1		0
	0	1	...	27	...	'A'	...	'E'	...	'I'	...	'N'	...	'P'	...	'S'	...	'T'	...	255

n = 7 + 1

HUFFMAN: kódolás/dekódolás

Implementációs ötletek (2-3-4-5)

huffman															
apa															
balfiu															
jobbfiu															
frekvencia	1	2	1	2	1	1	1	1							
karakter	esc	A	E	I	N	P	S	T							
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

```
typedef struct{
    int apa, balfiu, jobbfiu, frekvencia;
    char karakter;
} ELEM;

ELEM *huffman = (ELEM*)calloc(2*n-1, sizeof(ELEM));
```

huffman

apa	-8		8												
balfiu								0							
jobbfiu								2							
frekvencia	1	2	1	2	1	1	1	1	2						
karakter	esc	A	E	I	N	P	S	T							
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

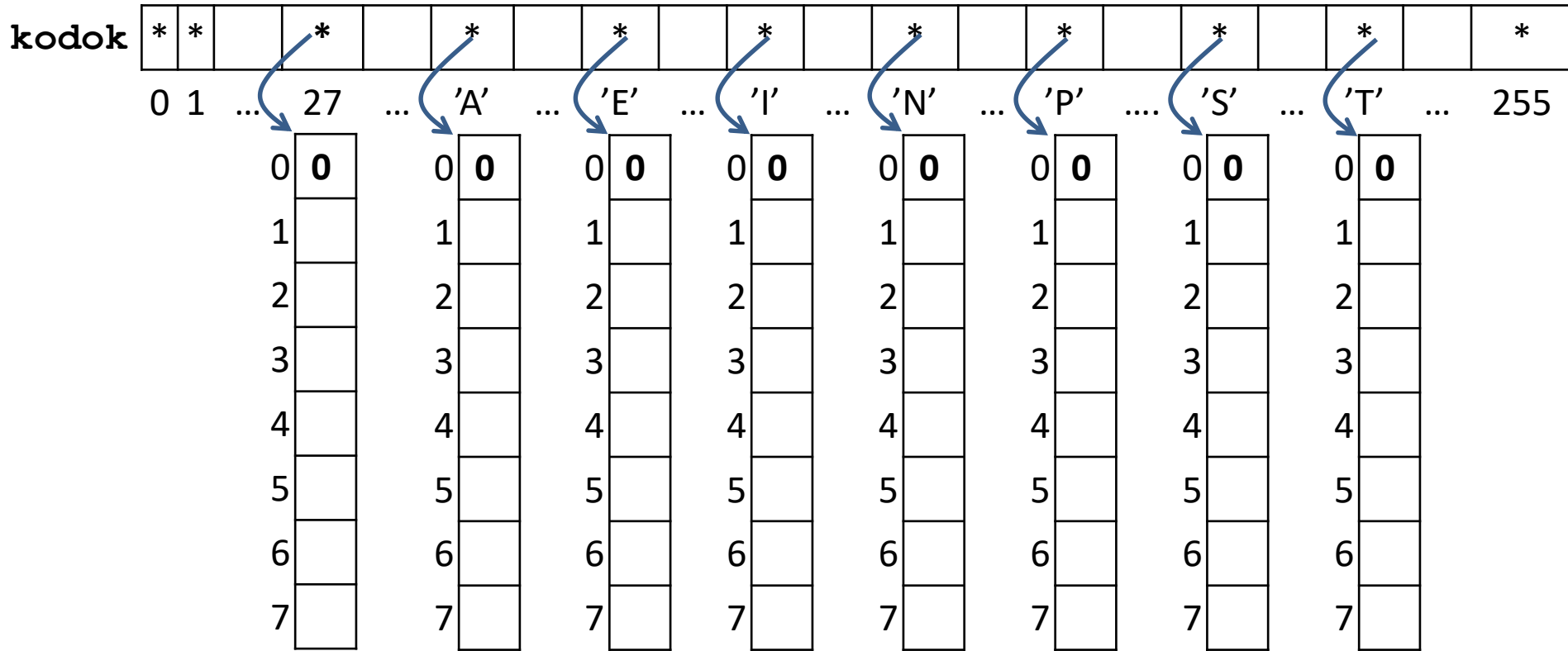
```
int m = n;
for ( int i = 0 ; i < n-1 ; ++i ) {
    int balfiu = minindex(huffman, m);
    huffman[balfiu].apa = (-1)*m;
    huffman[m].frekvencia += huffman[balfiu].frekvencia;
    huffman[m].balfiu = balfiu;
    int jobbfiu = minindex(huffman, m);
    huffman[jobbfiu].apa = m;
    huffman[m].frekvencia += huffman[jobbfiu].frekvencia;
    huffman[m].jobbfiu = jobbfiu;
    ++m;
}
```

huffman

apa	-8		8		-9	9									
balfiu								0	4						
jobbfiu								2	5						
frekvencia	1	2	1	2	1	1	1	1	2	2					
karakter	esc	A	E	I	N	P	S	T							
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

huffman

apa	-8	-11	8	11	-9	9	-10	10	-12	12	-13	13	-14	14	0
balfiu	0	0	0	0	0	0	0	0	0	4	6	1	8	10	12
jobbfiu	0	0	0	0	0	0	0	0	2	5	7	3	9	11	13
frekvencia	1	2	1	2	1	1	1	1	2	2	2	4	4	6	10
karakter	esc	A	E	I	N	P	S	T							
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14



```

char **kodok;
kodok = (char**)calloc(256, sizeof(char*));
for ( int i = 0 ; i < 256 ; ++i ){
    if( stat[i] ){
        kodok[i] = (char*)calloc(n,sizeof(char));
    }
}

```

huffman

apa	-8	-11	8	11	-9	9	-10	10	-12	12	-13	13	-14	14	0
balfiu	0	0	0	0	0	0	0	0	0	4	6	1	8	10	12
jobbfiu	0	0	0	0	0	0	0	0	2	5	7	3	9	11	13
frekvencia	1	2	1	2	1	1	1	1	2	2	2	4	4	6	10
karakter	esc	A	E	I	N	P	S	T							
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

```
void epitkod(ELEM *huffman, int id, char *kod){
    if( huffman[id].apa ){
        if(huffman[id].apa>0){
            epitkod(huffman, huffman[id].apa, kod);
            strcat(kod, "1");
        }
        else{
            epitkod(huffman, -huffman[id].apa, kod);
            strcat(kod, "0");
        }
    }
}
```

kodok

*	*		*		*		*		*		*		*		*		*		*
0	1	...	27	...	'A'	...	'E'	...	'I'	...	'N'	...	'P'		'S'	...	'T'	...	255
0	1		0	1		0	0		0	1		0	0		0	0		0	1
1	1		1	0		1	0		1	1		1	1		1	1		1	0
2	1		2	1		2	0		2	0		2	0		2	1		2	0
3	1		3	/0		3	/0		3	/0		3	/0		3	/0		3	/0
4	/0		4			4			4			4			4			4	

```
for ( int i = 0 ; i < n ; ++i ) {  
    epitkod(huffman, i, kodok[huffman[i].karakter]);  
}
```

Mohó algoritmusok (GREEDY)

- A greedy módszert optimalizálási feladatok megoldására használjuk.

(Felvonó) Egy egyszemélyes felvonó előtt n személy áll, akikről ismert, hogy hányadik emeletre szeretnének feljutni ($e_1, e_2, \dots, e_n$). Milyen sorrendben kellene használni a felvonót, ha azt szeretnék, hogy a várakozási idejük összege minimális legyen? Mennyi lesz ez az összidő, ha tudjuk, hogy a felvonó időegységeként egy emeletmagasságot tesz meg, és a kiszállás és beszállás „szempillantás alatt” történik?

(Műsorok) Adott n tévéműsor, amelyeknek ismert a kezdési és befejezési időpontjuk: $(b_1, e_1), (b_2, e_2), \dots, (b_n, e_n)$. Egy család, amelynek egy tévékészüléke van, úgy dönt, hogy a $[B, E]$ időintervallumban $(b_i \geq B, e_i \leq E, i = 1, n)$ tévézni fog. Mely műsorokat válasszák (lehetnek átfedő műsorok is, amelyeket természetesen más-más kanálison közvetítenek), ha azt szeretnék, hogy a legtöbb műsort lássák? Legkevesebb hány tévékészülékre lenne szükségük (és legalább hány tagú kellene hogy legyen a család) ahhoz, hogy minden műsort megnézhessen legalább egy családtag?

(Telefonhálózat) N számú város között telefonhálózatot szeretnének kiépíteni. Egy $d[1..m]$ egydimenziós tömbben adott, hogy mely várospárok között építhető ki direkt telefonvonal, valamint, hogy ezek a kapcsolatok egyenként mennyibe kerülnének. Az i -edik közvetlen vonal végpontjait, valamint megépítési költségét a $d[i].x$, $d[i].y$ és $d[i].k$ mezők tárolják. Mely városok között építsék ki a közvetlen telefonvonalakat ahhoz, hogy összekapcsoljanak minden várost (legalább közvetve), és a költségek minimálisak legyenek?

(*Madarak*) Adott $n + 1$ fa, amelyek 1-től $(n + 1)$ -ig vannak megszámozva. Az első n fa mindenikén elhelyezkedett egy-egy madár. Ezeket is megszámozzuk 1-től n -ig. Az i -edik fára az i -edik madár szállt ($i = 1, n$). A madarak elkezdenek áthelyezkedni. Minden lépésben valamelyik madár átrepül az éppen üres fára (egyszerre csak egyetlen madár van a levegőben). Ismerve, hogy egy idő után melyik madár éppen melyik fára szállt, „repítsük vissza” a madarakat eredeti helyükre *minimális* számú repüléssel.

(*Legrövidebb utak*) Adott egy $n \times n$ méretű d mátrix, amely egy n várost összekötő úthálózatot ábrázol. A $d[i][j]$ elem az i és j városok közti közvetlen út hosszát tárolja (ha két város közt nincs direkt út, a megfelelő mátrixelemek értéke ∞). Határozzuk meg az első várostól az összes többihez vezető *legrövidebb* utakat és ezeknek a hosszát.

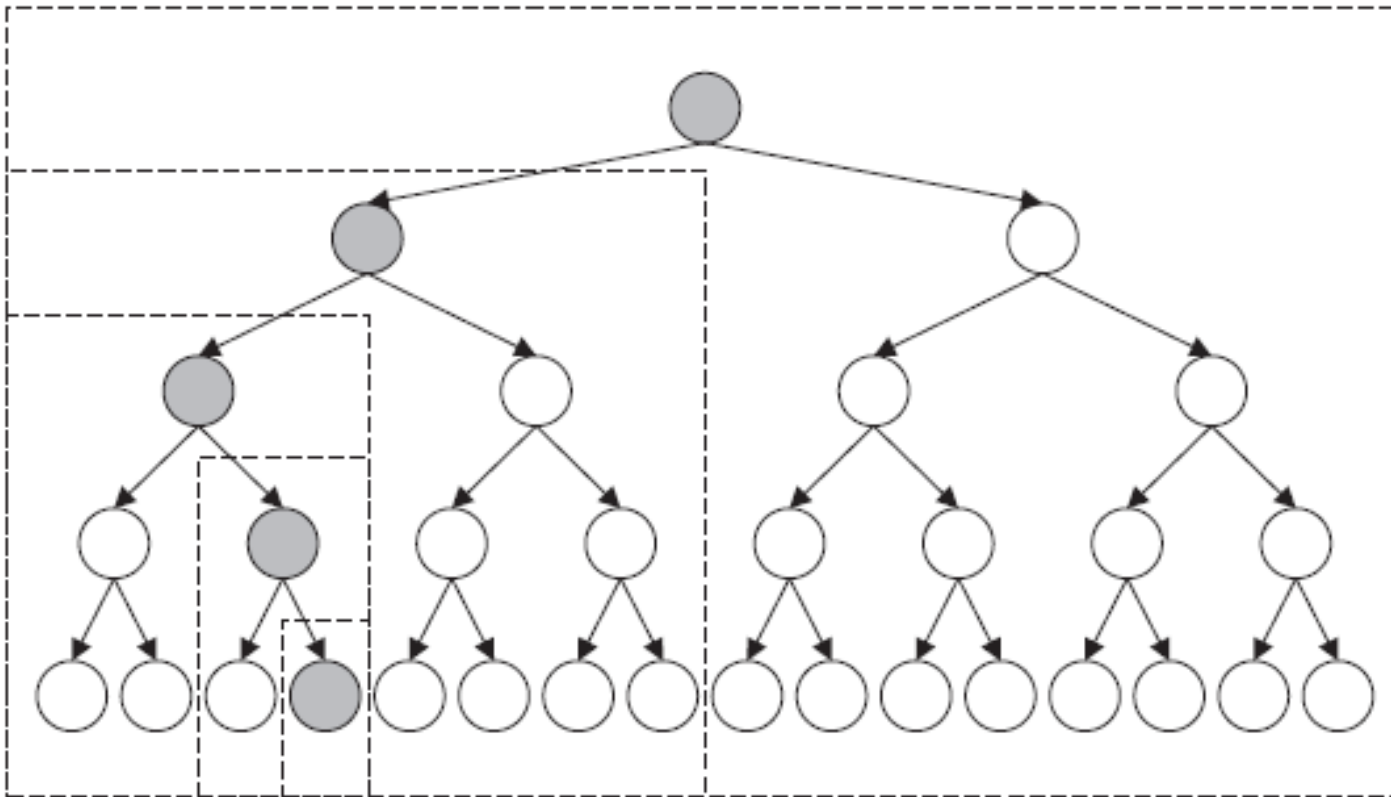
Mohó stratégia

Mindent feltesz egy lapra!

- Mindig az adott lépésben optimálisnak látszó (legígéretesebb) döntést hozzuk
 - Nem számolva az esetleges hosszú távú következményekkel
- Úgy gondolkodunk, hogy a lokális optimum majd globális optimumhoz vezet
 - Ez általánosságban nem igaz
 - Mohó melléfogások:
 - Nem az optimális megoldást találja meg
 - 6 euró kifizetése minimális számú 1,3,4 eurósokkal
 - Úgy érzékeli nincs megoldás
 - 9 euró kifizetése minimális számú 3,4 eurósokkal

Mohó stratégia a fán

- Minden döntéssel a feladat *egy* hasonló egyszerűbbé redukálódik
 - Optimális megoldás = optimális gyökér-levél út



Mohó döntések

1. Mivel annak a személynek a feljutási ideje, aki éppen használja a felvonót, bekerül a többi, még sorára váró lakó várakozási idejébe is, ezért mindig a *legalacsonyabban lakó* személy lesz a következő.
2. A *leghamarabb befejeződő* műsort választjuk ki elsőként, hogy a lehető leghosszabb *folytonos* időszakasz maradjon fenn további választásokra. Ha a soron következő legígéretesebb műsor átfedődik a már kiválasztottakkal, egyszerűen kihagyjuk.
3. Mindig a következő *legolcsóbb* szakaszon építjük ki a telefonvonalat, ügyelve arra, hogy ne építsünk közvetlen vonalat ott, ahol már létrejött közvetett kapcsolat.
4. Arra törekszünk, hogy mindenik madár a lehető *legkevesebb* repüléssel kerüljön a helyére. Tehát mindig az a madár fog repülni, amelyiknek a fája éppen üres. Ily módon az illető madár egy repüléssel a helyére kerül. Ha egy adott pillanatban az $(n + 1)$ -edik fa válik üressé, és nincs még minden madár a helyén, akkor ezek közül valamelyik elrepül az $(n + 1)$ -edik fára. Ez a madár végül két repülésből kerül a fájára.
5. Mindig a már elért városokhoz *legközelebb* eső város lesz a következő, amelyikhez meghatározzuk a legrövidebb utat. Ez a sorrend biztosítja majd, hogy minden városhoz a legrövidebb úton jussunk el.

Kandidátusok halmaza

Megoldás halmaz

Mohó – váz

eljárás greedy(K)

$S \leftarrow \Phi$

amíg nem megoldás(S) és $K \neq \Phi$

$x \leftarrow \text{legígéretesebb}(K)$

$K \leftarrow K \setminus \{x\}$

ha bővíthető(S,x) akkor

$S \leftarrow S \cup \{x\}$

vége ha

vége amíg

ha megoldás(S) akkor

kiír(S)

különben

ki: "Nincs"

vége ha

vége greedy

Például (lásd 2. feladatot):
a következő legkorábban
befejeződő műsor

(1)

(2)

(3)

Nem fedi át a már programra
tűzött műsorokat

- (1) – mindig az adott pillanatban legígéretesebbnek látszó kandidátust választjuk (a döntési fa aktuális csomópontjának legígéretesebb fiát);
- (2) – a kiválasztott kandidátust „örökre” eltávolítjuk a kandidátushalmazból;
- (3) – ha a megoldás halmaz bővíthető az illető elemmel, akkor végérvényesen beletesszük, ha nem, akkor végképp lemondunk róla.

Műsorok feladat megoldása

Ígéretességi sorrendbe rendezzük a műsorokat (a kandidateket)

```
műsorok (e[],b[],n)
  rendezés(e,b,n) // az e és b tömbök párhuzamos rendezése
                  // a befejezési időpontok szerint
  ki: '(',b[1],',',',e[1],')'
  V=e[1]
  minden i=2,n végezd
    ha b[i] > V akkor
      ki: '(',b[i],',',',e[i],')'
      V = e[i]
    vége ha
  vége minden
vége műsorok
```

A soron következő műsor később kezdődik, mint az eddig programra tűzöttek

Matematikai háttér

- Mohó-választás alapelve
 - A feladat optimális megoldása mohó-választással kezdődik (vagy módosítható úgy, hogy mohó-választással kezdődjön),
 - és ezen választás nyomán a feladat hasonló feladattá redukálódik.
- Optimálitás alapelve
 - A feladat optimális megoldása a részfeladatok (amelyekre a feladat a mohó-döntések nyomán redukálódik) optimális megoldásaiból épül fel.

Élj a mának?!

„Minden napnak elég a maga baja”

„Amit vet az ember, azt fogja aratni is”