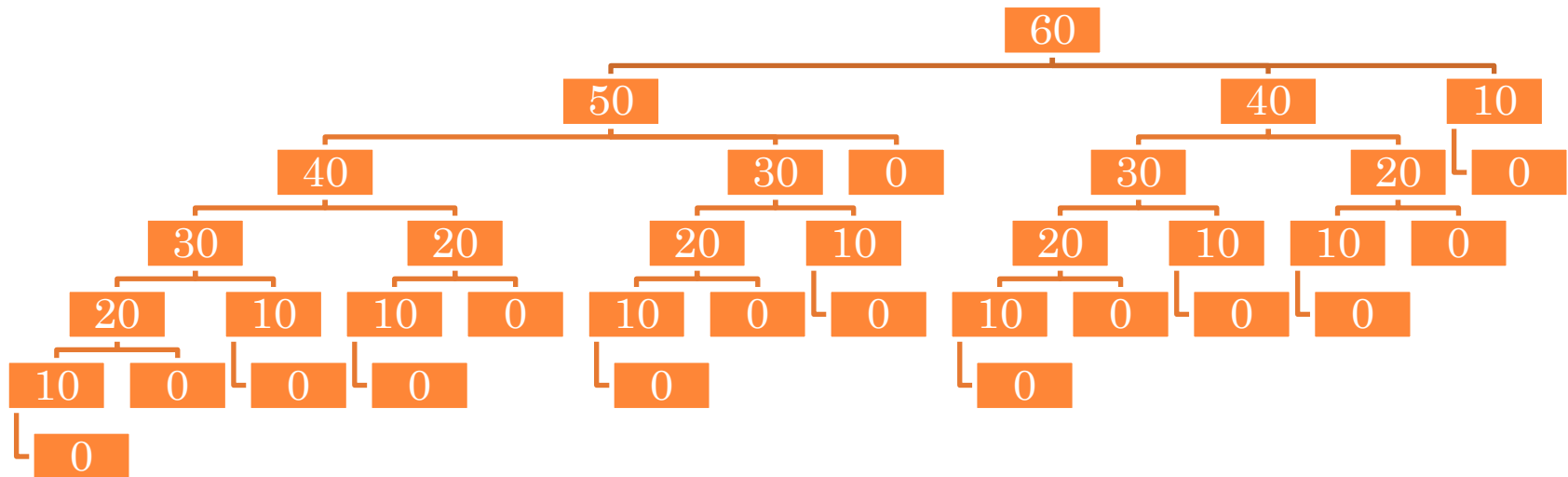




PAŽRAVÉ ALGORITMY (GREEDY) A BACKTRACKING

PROBLÉM MINCOVKA

- rozmeňme danú sumu peňazí čo najmenším počtom mincí 10c, 20c, 50c
- strom všetkých možností rozmenenia sumy 60c:

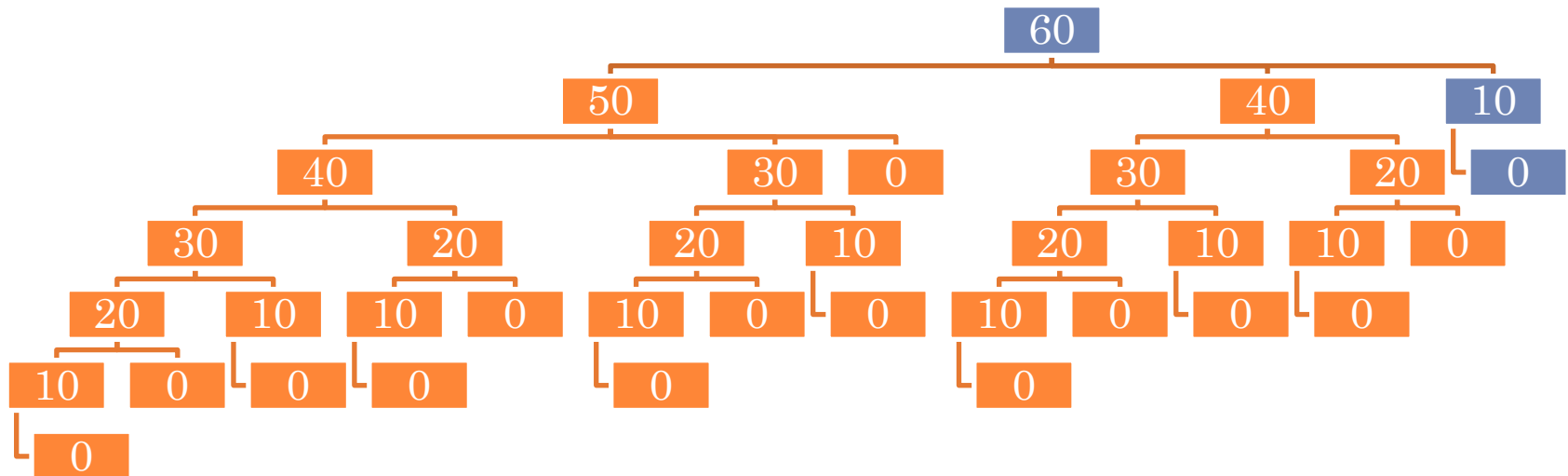


- vrcholy výpočtového stromu obsahujú sumu, ktorú treba rozmeniť, z toho vyplýva, že v listoch je 0
- rozdiel medzi rodičom a potomkom je hodnota vyplatenej mince
- hľadáme najkratšiu cestu v strome od koreňa k listu

PROBLÉM MINCOVKA – PAŽRAVÝ ALGORITMUS

- v každom kroku výpočtu vyberieme najväčšiu možnú mincu, ktorá sa dá použiť (v poradí 50c, 20c, 10c)

$$60c = 50c + 10c$$



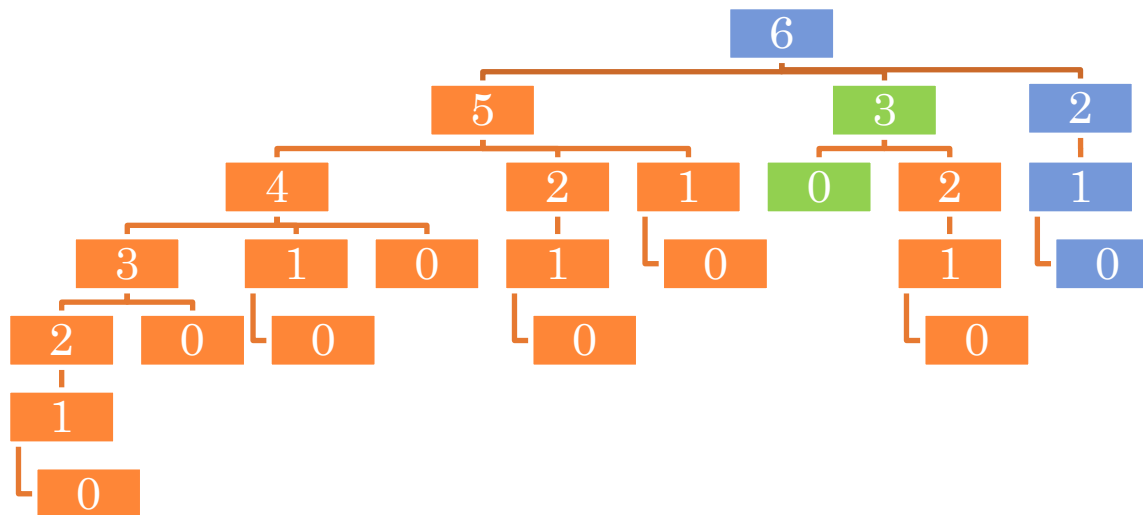
PROBLÉM MINCOVKA – PAŽRAVÝ ALGORITMUS

- rozmeňme danú sumu peňazí čo najmenším počtom mincí 2€, 1€, 50c, 20c, 10c, 5c, 2c, 1c

$$9,84\text{€} = 2\text{€} + 2\text{€} + 2\text{€} + 2\text{€} + 1\text{€} + 50\text{c} + 20\text{c} + 10\text{c} + 2\text{c} + 2\text{c}$$

NESPRÁVNE RIEŠENIE PAŽRAVÝM ALGORITMOM:

- rozmeňme mincami 1, 3, 4 sumu 6
- pažravý algoritmus nájde: $6 = 4 + 1 + 1$
- optimálne riešenie je: $6 = 3 + 3$



PAŽRAVÝ (GREEDY) ALGORITMUS

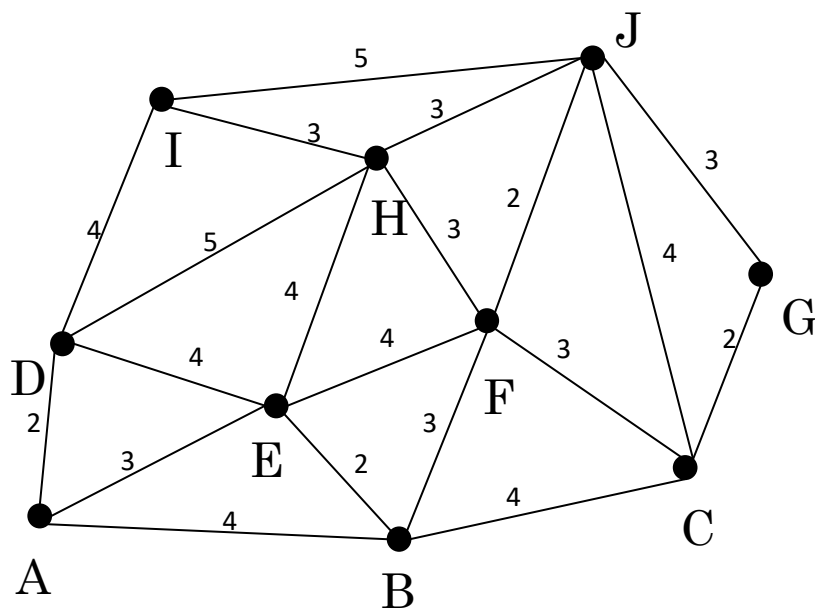
- z viacerých možností sa vyberá **lokálne** najlepšia
- nie vždy vedie k optimálnemu riešeniu problému, napr. majme mince s hodnotami: 1, 7, 10

Rozmeňme sumu 14

- pažravým algoritmom: 10, 1, 1, 1, 1,
- optimálne riešenie: 7, 7
- ak áno, čas je oveľa lepší ako pri prehľadávaní všetkých možností
 - n je suma na rozmenenie, k je počet mincí
 - pažravý algoritmus $O(n)$
 - prehľadávanie všetkých možností $O(k^n)$

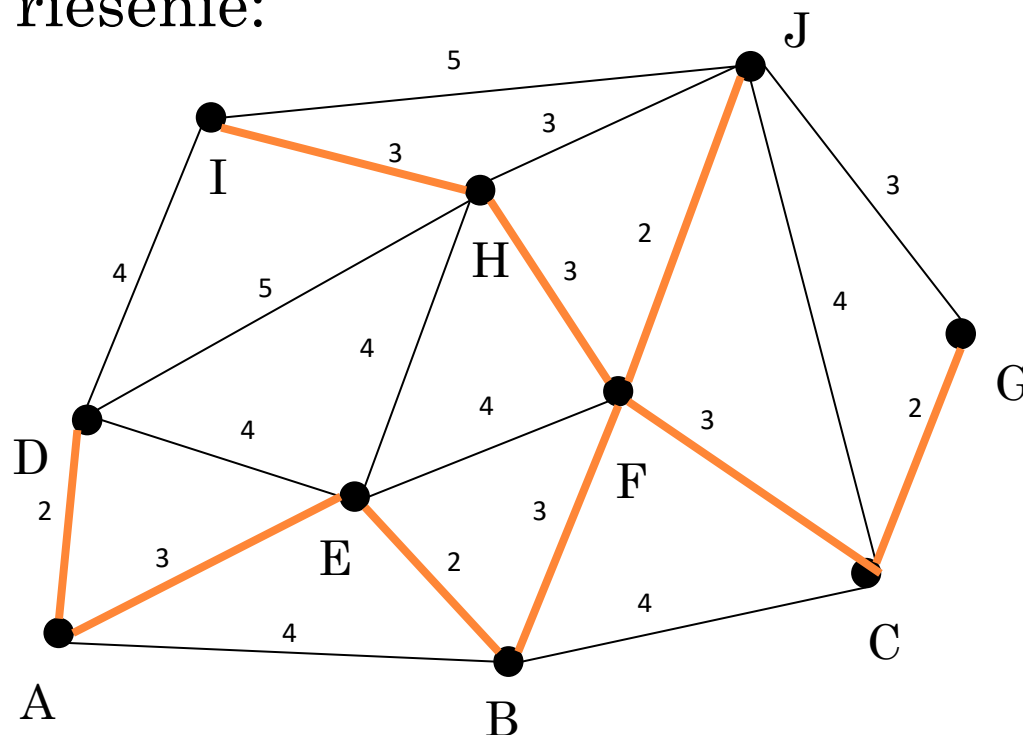
MINIMÁLNA KOSTRA GRAFU

- Vydĺáždime chodníky v meste tak, aby existovala cesta z každého domu do každého a dĺžka vydĺáždených chodníkov bola čo najkratšia.



PAŽRAVÝ ALGORITMUS 1 (KRUSKALOV)

- Dláždime cesty od najkratších k najdlhším. Ak je cesta zbytočná, t.j. spája domy, ktoré sú už spojené, zostáva nevydláždená.
- Možné riešenie:



KRUSKALOV ALGORITMUS

Vstup: graf $G=(V,E)$ s ohodnotenými hranami daný zoznamom hrán, $|V| = n$, $|E| = m$

Výstup: minimálna kostra – množina hrán $T \subset E$ taká, že T je strom a súčet ohodnotení hrán je minimálny

$T \leftarrow \emptyset$

usporiadať $E=\{ e_j, j=1..m \}$ podľa ohodnotenia hrán vzostupne;

for each $v_i \in V$ **do** $\text{komponent}[v_i] \leftarrow i$;

$j \leftarrow 0$;

while $|T| < n - 1$ **do begin**

$j \leftarrow j + 1$;

if $j > m$ **then break**;

$(u,v) \leftarrow e_j$;

if $\text{komponent}[u] \neq \text{komponent}[v]$ **then begin**

$T \leftarrow T \cup \{ e_j \}$;

for each $w \in V$ **do**

if $\text{komponent}[w] = \text{komponent}[v]$ **then**

$\text{komponent}[w] \leftarrow \text{komponent}[u]$

end;

end;

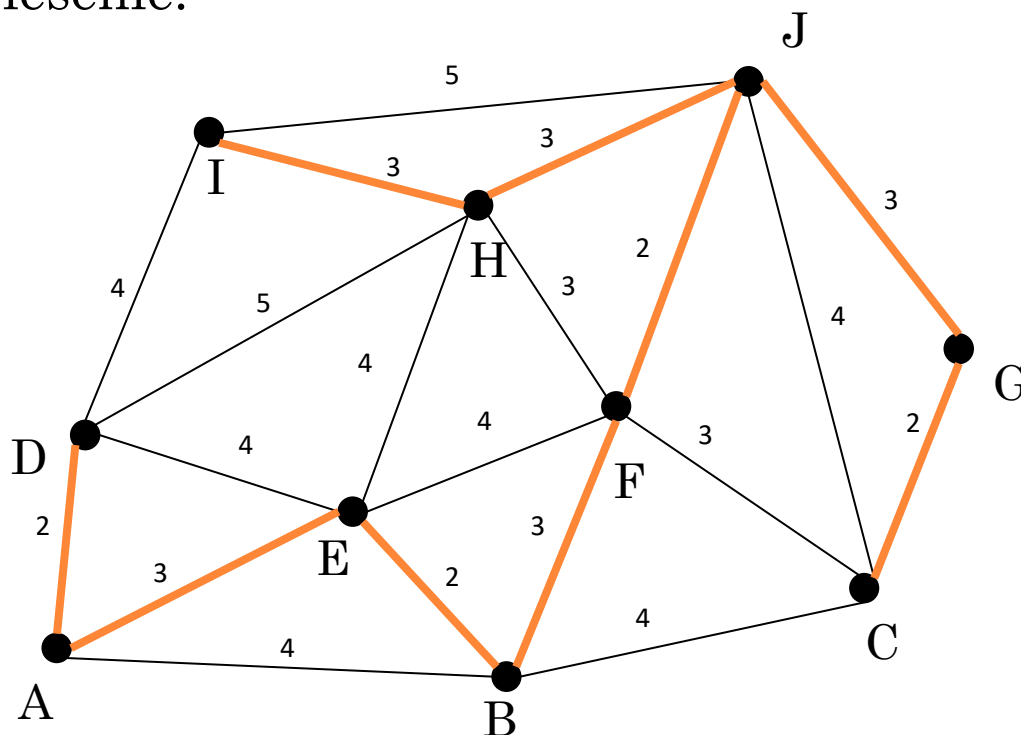
KRUSKALOV ALGORITMUS

- časová výpočtová zložitosť – závisí od implementácie grafu a komponentov súvislosti
- pre graf daný zoznamom m hrán s najviac n komponentmi súvislosti implementovanými poľom:
 - usporiadanie hrán: $O(m \log m)$
 - inicializácia komponentov súvislosti pre všetky vrcholy: $O(n)$
 - pridávanie hrán do kostry T : $O(m \cdot n)$

$$T(n, m) = O(m \cdot n)$$

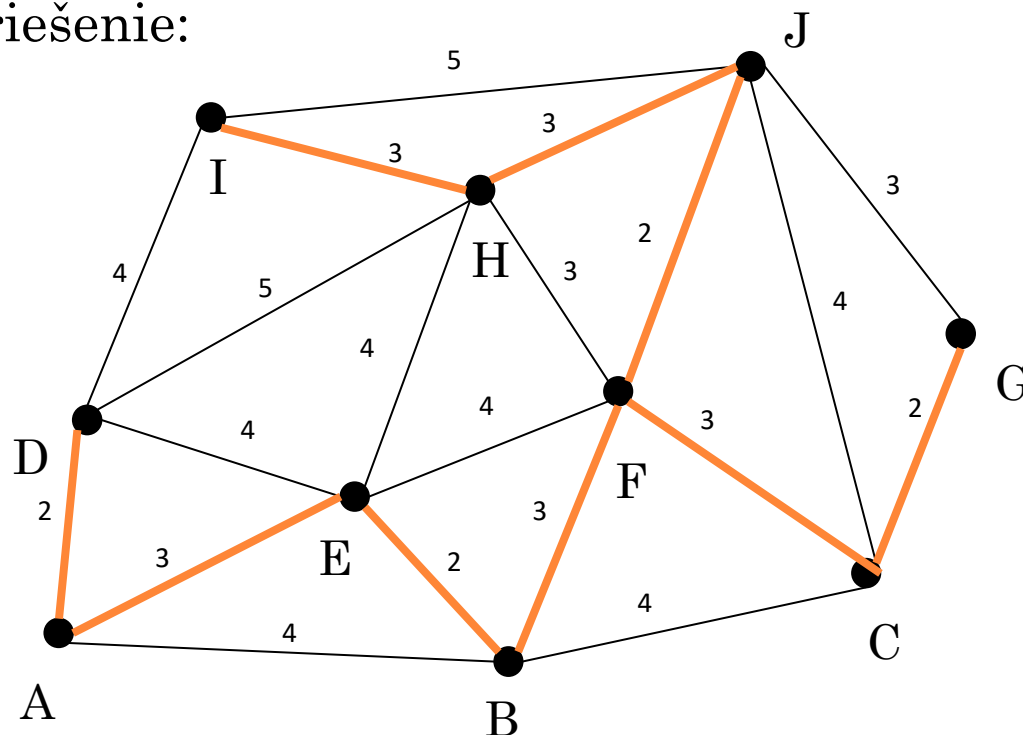
PAŽRAVÝ ALGORITMUS 2 (PRIMOV)

- Začneme z ľubovoľného domu. K nemu pripojíme najbližšieho suseda. Spomedzi nepripojených domov pripájame vždy ten, ktorý vieme pripojiť najkratšou cestou k minimálnej kostre.
- Možné riešenie:



PAŽRAVÝ ALGORITMUS 3

- Všetky cesty prehlásime za vydláždené. Nájdeme najdlhšiu cestu, ktorá nerozpojí cestnú sieť a tú zrušíme. Postup opakujeme, kým nezostane o jednu menej ciest ako domčekov.
- Možné riešenie:




VÝPOČTOVÁ ZLOŽITOSŤ PAŽRAVÝCH ALGORITMOV

- Pažravý algoritmus typicky robí $O(n)$ výberov pri riešení problému rozsahu n
- Nech $výber(n)$ je zložitosť výberu z n objektov
- Zložitosť pažravého algoritmu je $O(n \cdot O(výber(n)))$
- mincovka:
 - n je vstupná suma peňazí, k je počet mincí
 - zložitosť výberu najväčšej mince použitej na vyplatenie je $O(k)$, čo je $O(1)$, lebo k je konštanta
 - zložitosť algoritmu je $O(n)$

ČO JE NEDETERMINISTICKÝ ALGORITMUS

- Príklad: Prejdime všetky polia šachovnice šachovým koňom tak, že každé pole navštíví práve raz.
- Riešenie:

20	17	4	27	46	15	2	29
5	26	19	16	3	28	45	14
18	21	32	47	44	13	30	1
25	6	43	34	31	62	49	56
22	33	24	53	48	55	12	63
7	38	35	42	61	52	57	50
36	23	40	9	54	59		11
39	8	37	60	41	10	51	58

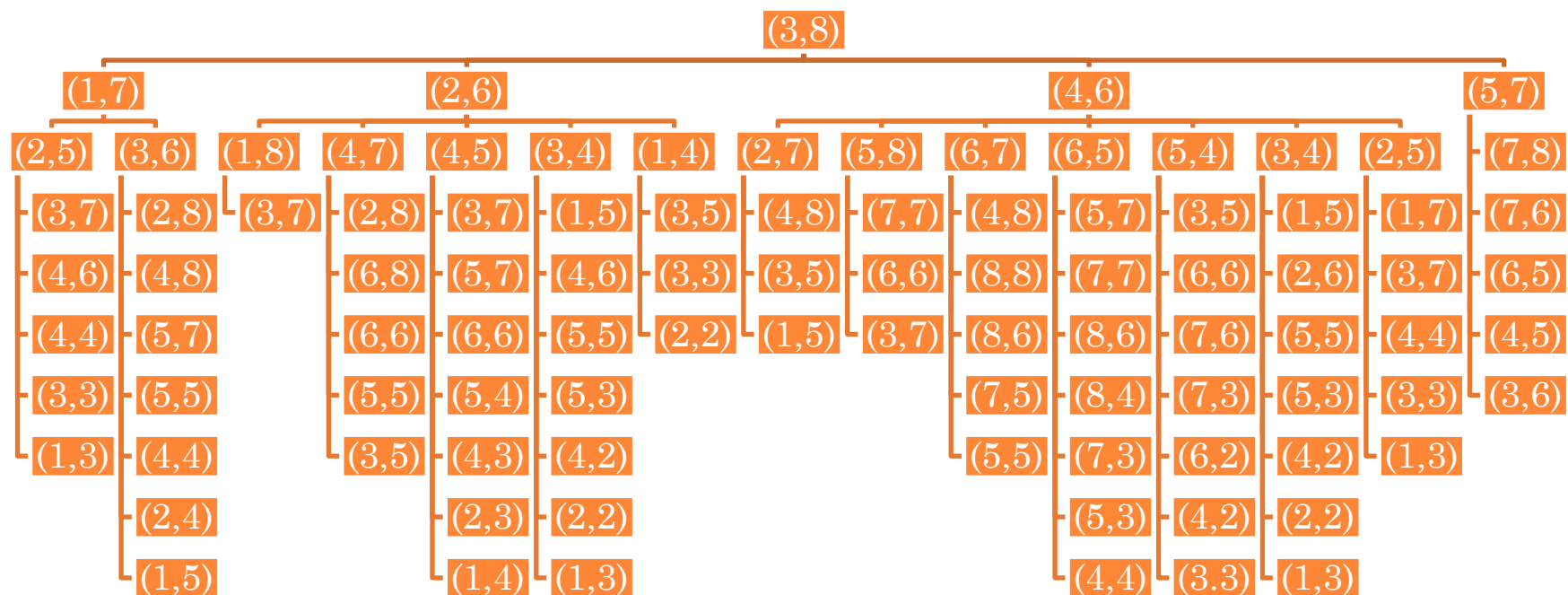
NEDETERMINISTICKÝ ALGORITMUS

rekurzívna procedúra **skok(x,y)**:

```
ak sú obsadené všetky polia šachovnice, tak:  
    nájdené riešenie  
inak  
    ak existuje prázdne pole (x', y') dosiahnuteľné jedným  
    skokom koňa, tak:  
        obsad' ho  
        skok(x', y')  
    inak  
        riešenie neexistuje
```

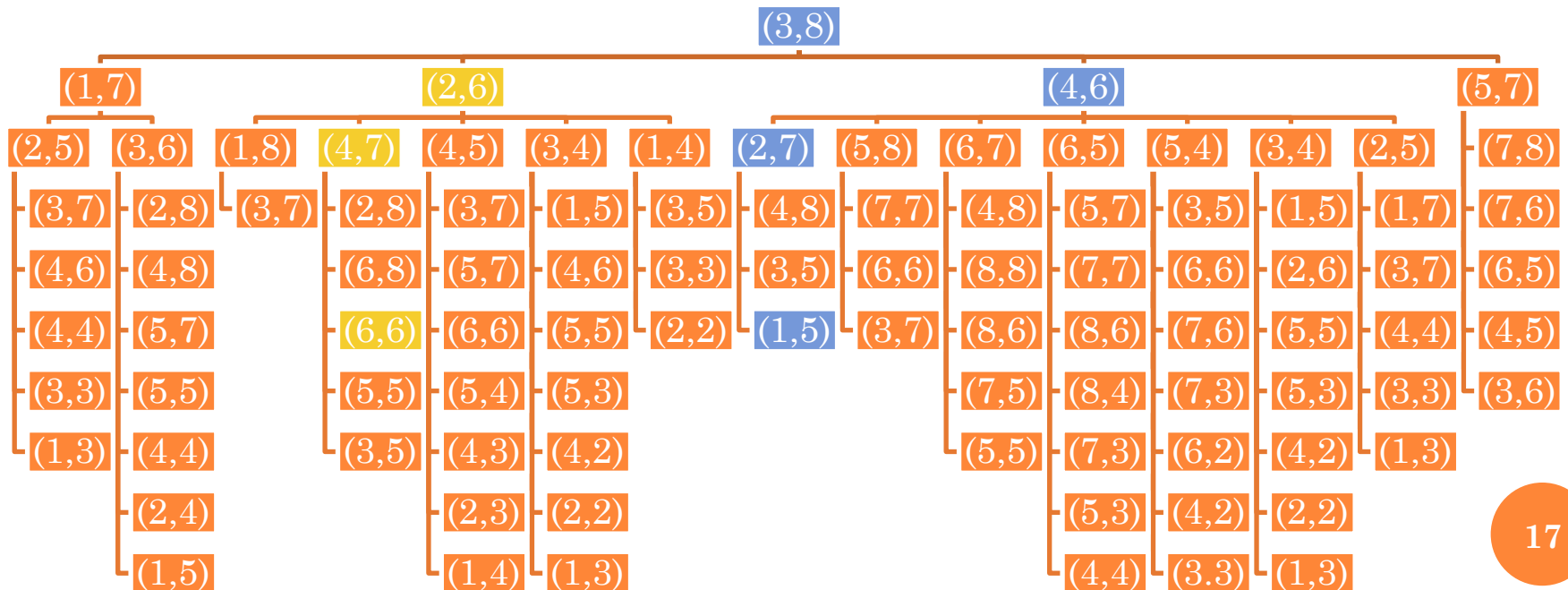
- polí (x', y') môže byť viac
- nedeterministický algoritmus môže generovať pre jeden vstup viac možných výpočtov
- možné výpočty znázorňuje výpočtový strom, v ktorom cesta od koreňa k listu predstavuje jeden možný výpočet

VÝPOČTOVÝ STROM – NEÚPLNÝ



VÝPOČTOVÝ STROM – NEÚPLNÝ

- výpočet nedeterministického algoritmu – cesta od koreňa k listu (hľadáme cestu dĺžky 64)
- dĺžka nedeterministického výpočtu – hĺbka stromu
- deterministický algoritmus – prehľadanie stromu, dĺžka výpočtu – súčet dĺžok všetkých ciest



BACKTRACKING – PREHLADÁVANIE S NÁVRATOM

- prehľadávanie výpočtového stromu do hĺbky
- procedúra na prehľadanie podstromu s koreňom vo vrchole V: **prehľadaj(V)**

ak V je cieľový vrchol, tak
 nájdene riešenie
 zastav prehľadávanie
inak
 ak V je list, tak
 nenájdene riešenie
 inak
 pre každý potomok P vrcholu V:
 prehľadaj(P)

- Výpočtová zložitosť prehľadávania:
 - $T(n) = O(c^{f(n)})$, kde c je konštanta – stupeň stromu, $f(n)$ je dĺžka najdlhšej vetvy (zložitosť nedeterministického algoritmu)

BACKTRACKING

- systematicky simuluje všetky možné výpočty nedeterministického algoritmu
- ak existuje výpočet, ktorý vedie k riešeniu problému, nájde ho
- výpočtová zložitosť nedeterministického algoritmu je hĺbka výpočtového stromu (dĺžka najdlhšej vetvy)
- výpočtová zložitosť deterministického algoritmu je súčet výpočtových zložítostí všetkých možných výpočtov nedeterministického algoritmu –
exponenciálna

FUNKCIE ZLOŽITOSTI PRE $n = 10, 100, 1000$

	10	100	1000
$O(1)$	1	1	1
$O(\log n)$	3,32	6,64	9,97
$O(n)$	10	100	1000
$O(n \log n)$	33,22	664,39	9965,78
$O(n^2)$	100	10000	1000000
$O(n^3)$	1000	1000000	1000000000
$O(2^n)$	1024	1,26765E+30	1,0715E+301
$O(n!)$	3628800	9,3326E+157	veľa
$O(n^n)$	100000000000	1E+200	veľa

TRIEDY ZLOŽITOSTI

- Trieda PTIME (P) je množina problémov, ktoré sa dajú riešiť deterministickým algoritmom v polynomiálnom čase $O(n^{\text{konšt}})$.
- Trieda NPTIME (NP) je množina problémov, ktoré sa dajú riešiť nedeterministickým algoritmom v polynomiálnom čase $O(n^{\text{konšt}})$.

Polynomiálna funkcia:

$$p(n) = a_0 + a_1n + a_2n^2 + \dots + a_cn^c = O(n^c)$$

ČO JE NP ÚLOHA?

- je taká úloha (problém), ktorá sa dá riešiť nedeterministicky v polynomiálnom čase
 - hĺbka výpočtového stromu závisí od vstupu polynomiálne
 - každá vetva stromu – nedeterministický výpočet – má polynomiálnu dĺžku
 - ak existuje riešenie problému, vieme **overiť jeho správnosť** deterministicky v polynomiálnom čase
 - na overenie neexistencie riešenia treba prehľadať celý výpočtový strom – v exponenciálnom čase