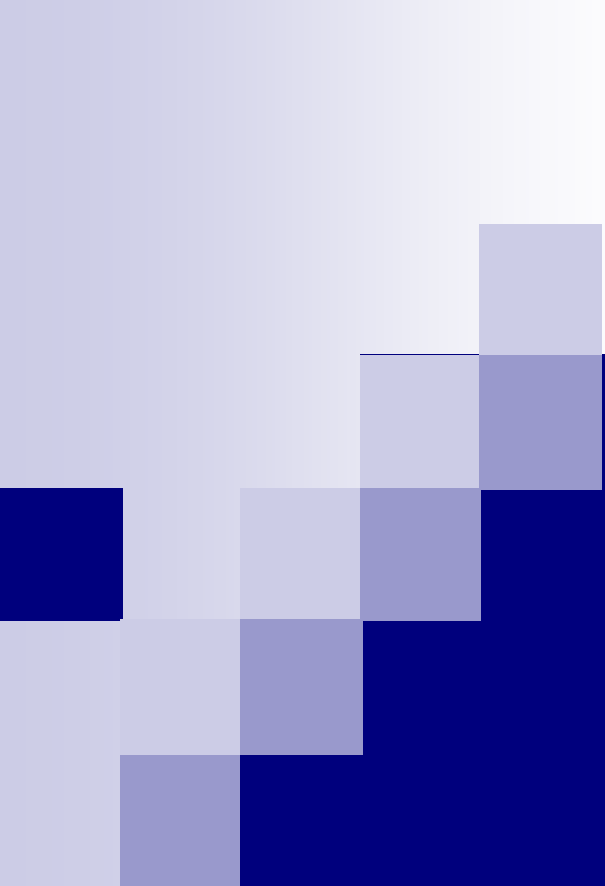


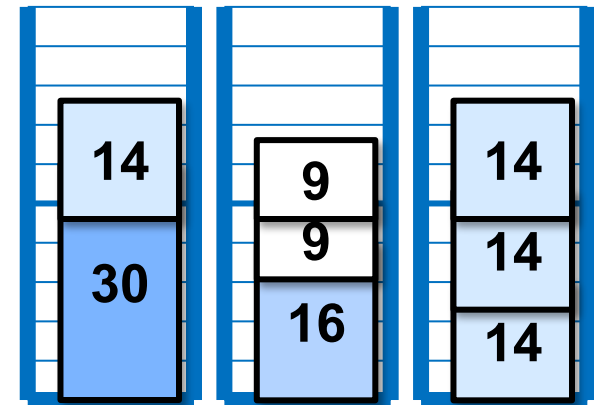
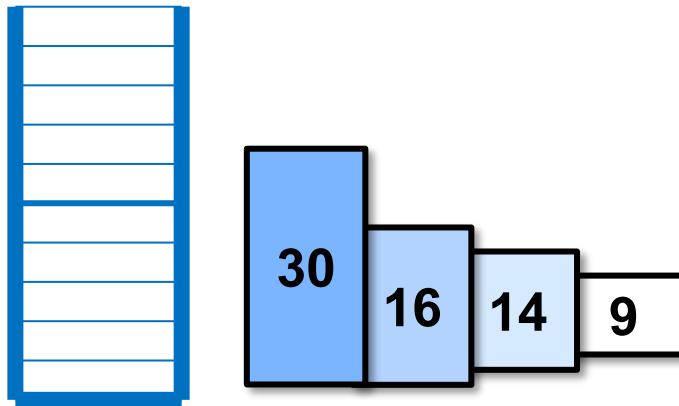


Algoritmizace

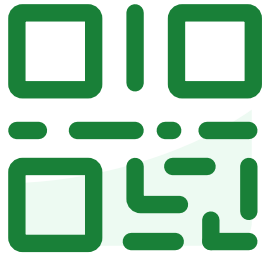
Daniel Průša, Robert Pěnička

2025

- Dynamické programování (DP)
 - Optimální binární vyhledávací strom
 - Úloha batohu neomezená
 - Úloha batohu 0/1



Join at slido.com #1189108

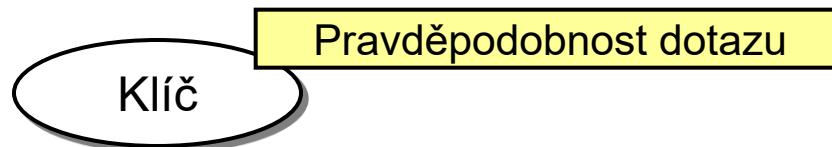
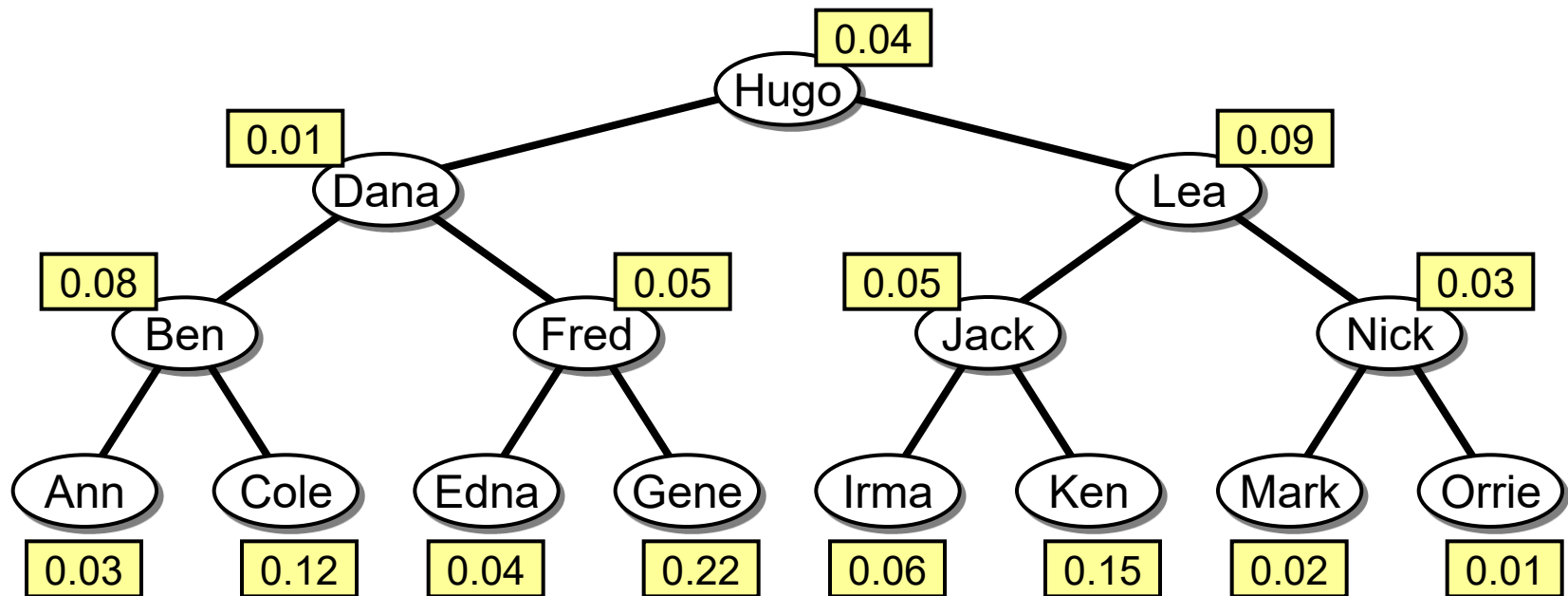


ⁱ The Slido app must be installed on every computer you're presenting from

slido

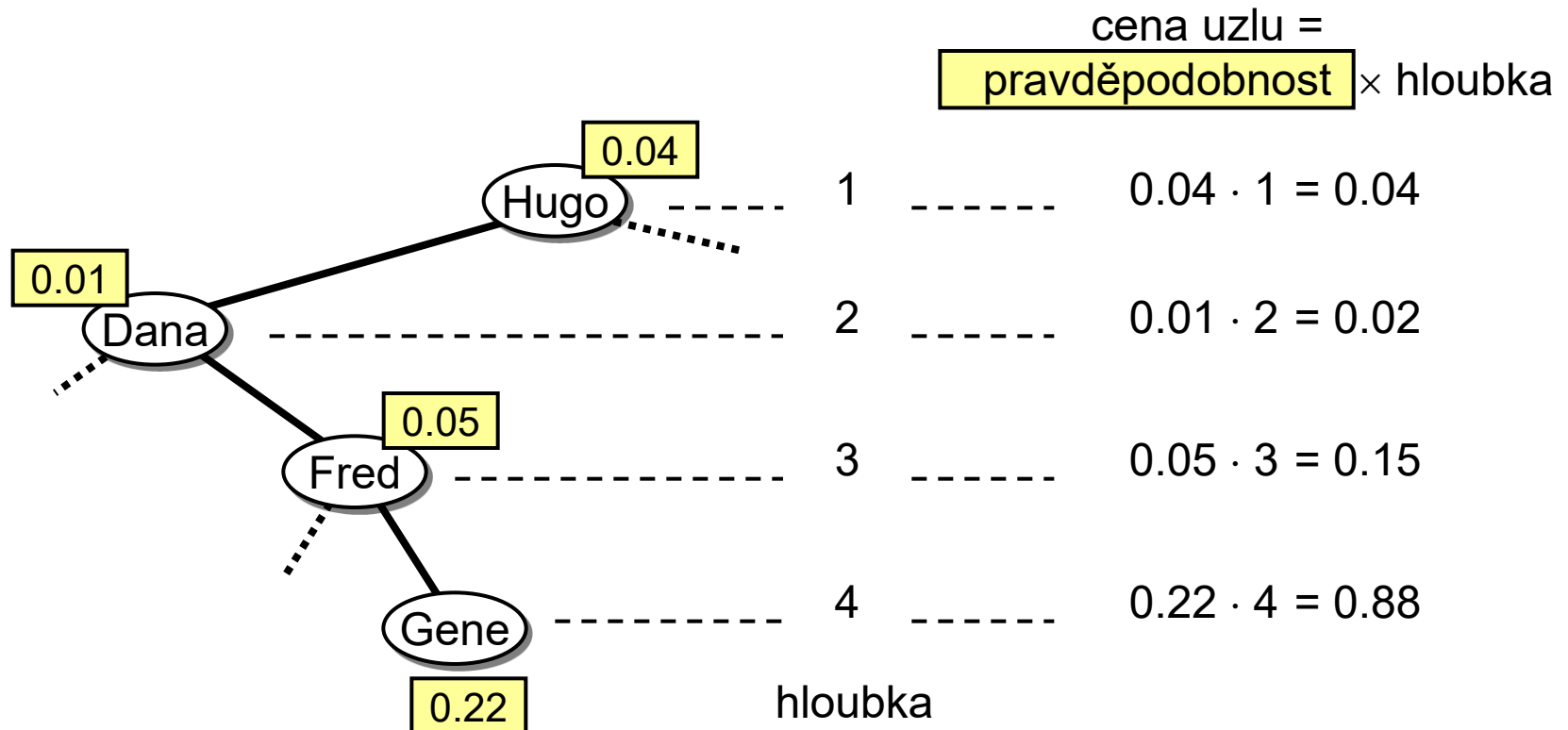
Optimální binární vyhledávací strom

Vyvážený, ale ne optimální



Optimální binární vyhledávací strom

Cena jednotlivých uzlů v BVS



cena uzlu = průměrný počet testů na nalezení uzlu
při jednom dotazu (Find)

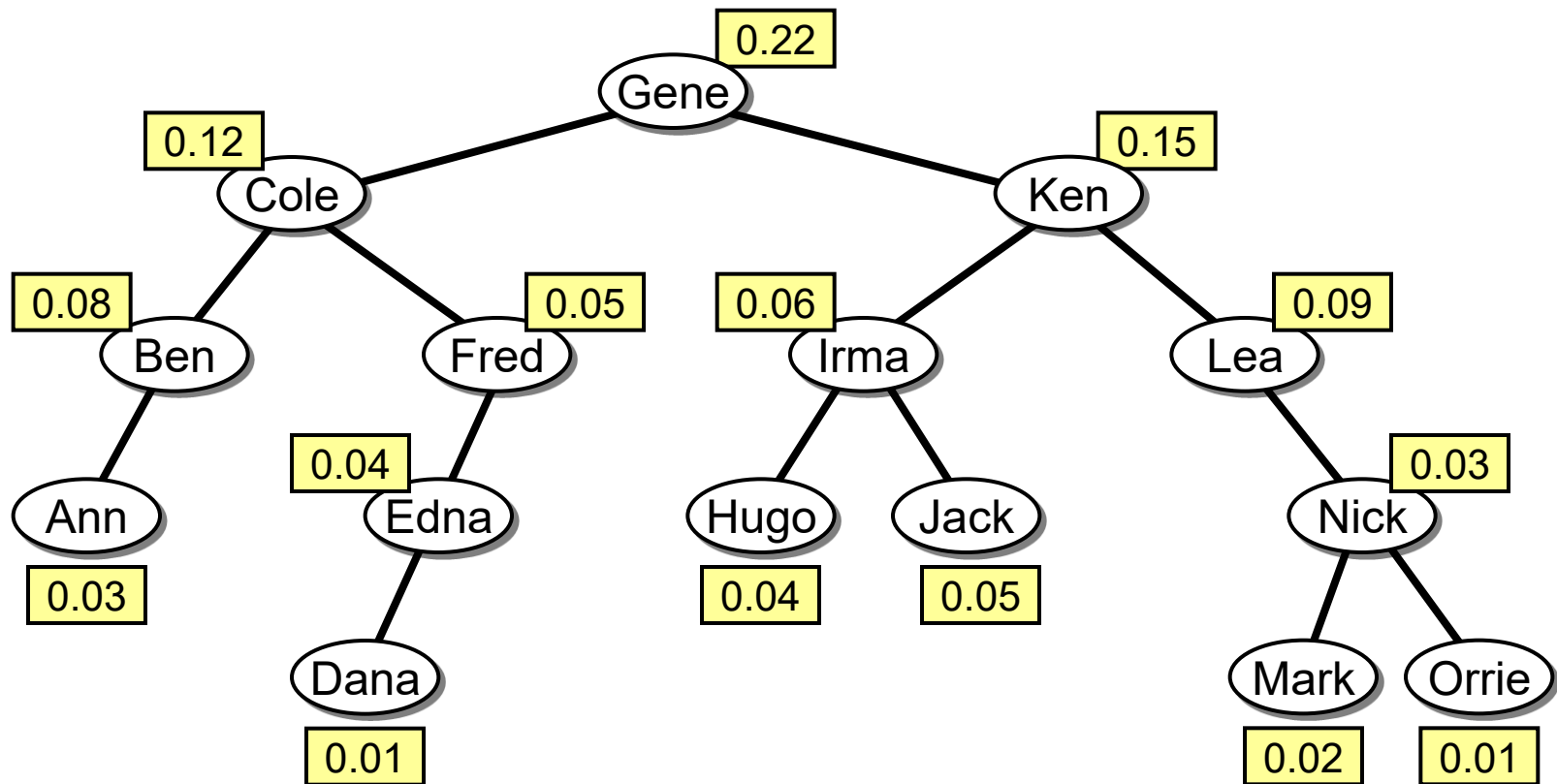
Cena vyváženého BVS

klíč	pravděp. p_k	hloubka d_k	$p_k \cdot d_k$
Ann	0.03	4	$0.03 \cdot 4 = 0.12$
Ben	0.08	3	$0.08 \cdot 3 = 0.24$
Cole	0.12	4	$0.12 \cdot 4 = 0.48$
Dana	0.01	2	$0.01 \cdot 2 = 0.02$
Edna	0.04	4	$0.04 \cdot 4 = 0.16$
Fred	0.05	3	$0.05 \cdot 3 = 0.15$
Gene	0.22	4	$0.22 \cdot 4 = 0.88$
Hugo	0.04	1	$0.04 \cdot 1 = 0.04$
Irma	0.06	4	$0.06 \cdot 4 = 0.24$
Jack	0.05	3	$0.05 \cdot 3 = 0.15$
Ken	0.15	4	$0.15 \cdot 4 = 0.60$
Lea	0.09	2	$0.09 \cdot 2 = 0.18$
Mark	0.02	4	$0.02 \cdot 4 = 0.08$
Nick	0.03	3	$0.03 \cdot 3 = 0.09$
Orrie	0.01	4	$0.01 \cdot 4 = 0.04$
Cena celkem:			3.47

Cena celkem = prům. poč. testů na jednu operaci Find.

Optimální binární vyhledávací strom

Struktura optimálního BVS s danými pravděpodobnostmi



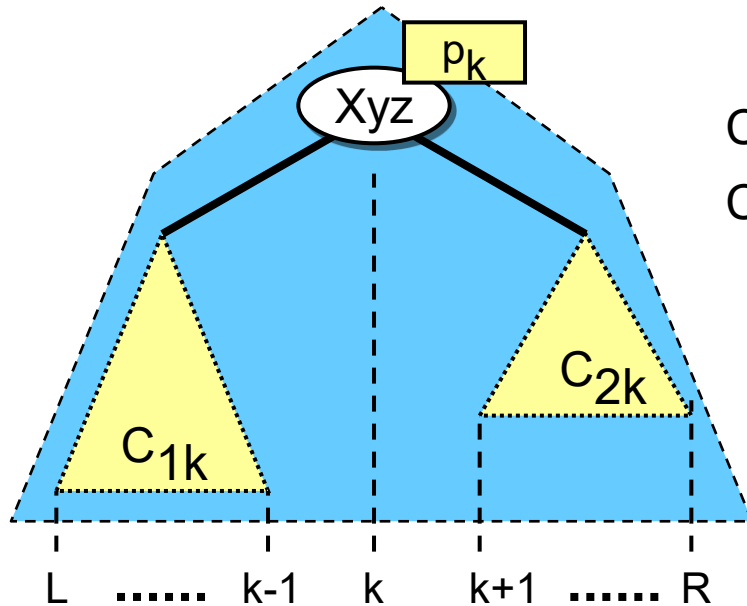
Cena optimálního BVS

klíč	pravděp. p_k	hloubka d_k	$p_k \cdot d_k$
Ann	0.03	4	$0.03 \cdot 4 = 0.12$
Ben	0.08	3	$0.08 \cdot 3 = 0.24$
Cole	0.12	2	$0.12 \cdot 2 = 0.24$
Dana	0.01	5	$0.01 \cdot 5 = 0.05$
Edna	0.04	4	$0.04 \cdot 4 = 0.16$
Fred	0.05	3	$0.05 \cdot 3 = 0.15$
Gene	0.22	1	$0.22 \cdot 1 = 0.22$
Hugo	0.04	4	$0.04 \cdot 4 = 0.16$
Irma	0.06	3	$0.06 \cdot 3 = 0.18$
Jack	0.05	4	$0.05 \cdot 4 = 0.20$
Ken	0.15	2	$0.15 \cdot 2 = 0.30$
Lea	0.09	3	$0.09 \cdot 3 = 0.27$
Mark	0.02	5	$0.02 \cdot 5 = 0.10$
Nick	0.03	4	$0.03 \cdot 4 = 0.12$
Orrie	0.01	5	$0.01 \cdot 5 = 0.05$
Cena celkem			2.56
Zrychlení	$3.47 : 2.56 = 1 : 0.74$		

Jak by šlo hledání optimálního BVS rozdělit na podúlohy?



Výpočet ceny optimálního BVS



C_{1k} cena levého podstromu uzlu k

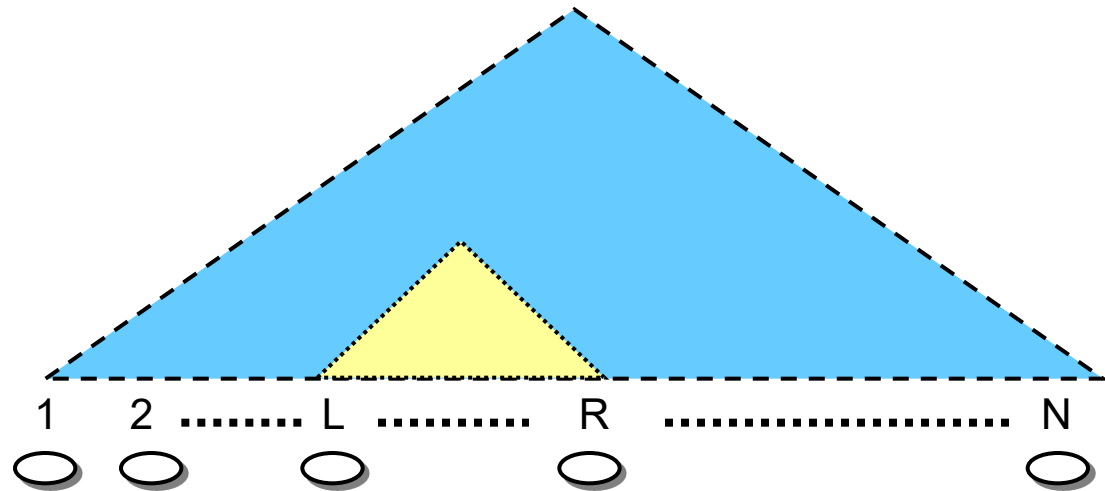
C_{2k} Cena pravého podstromu uzlu k

Rekurzivní myšlenka

$$\text{Cena} = C_{1k} + \sum_{i=L}^{k-1} p_i + C_{2k} + \sum_{i=k+1}^R p_i + p_k$$

Výpočet ceny optimálního BVS

Malé
optimální
podstromy



Nad prvky s indexy od L do R
lze jistě vytvořit jeden optimální podstrom.

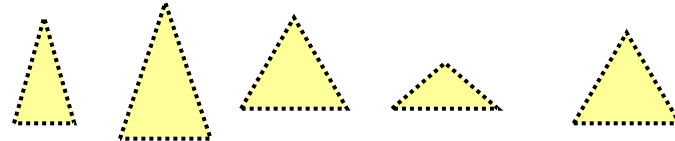
Velikost stromu
= poč. uzlů
= $L - R + 1$

Máme	N	optimálních podstromů velikosti	1
	N-1		2
	N-2		3
	⋮		⋮
	1	podstrom	N

Celkem máme $N * (N+1) / 2$ různých optimálních podstromů.

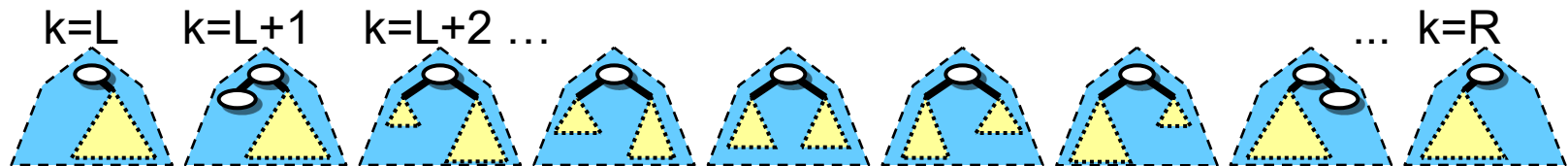
Minimalizace ceny BVS

Idea rekurzivního řešení:



1. Předpoklad : Všechny menší optimální stromy jsou známy.

2. Zkus: $k = L, L+1, L+2, \dots, R$



3. Zaregistruj index k , který minimalizuje cenu, tj. hodnotu

$$C_{1k} + \sum_{i=L}^{k-1} p_i + C_{2k} + \sum_{i=k+1}^R p_i + p_k$$

4. Klíč s indexem k je kořenem optimálního stromu.

Minimalizace ceny BVS

$C(L,R)$ Cena optimálního podstromu obsahujícího klíče
s indexy $L, L+1, L+2, \dots, R-1, R$

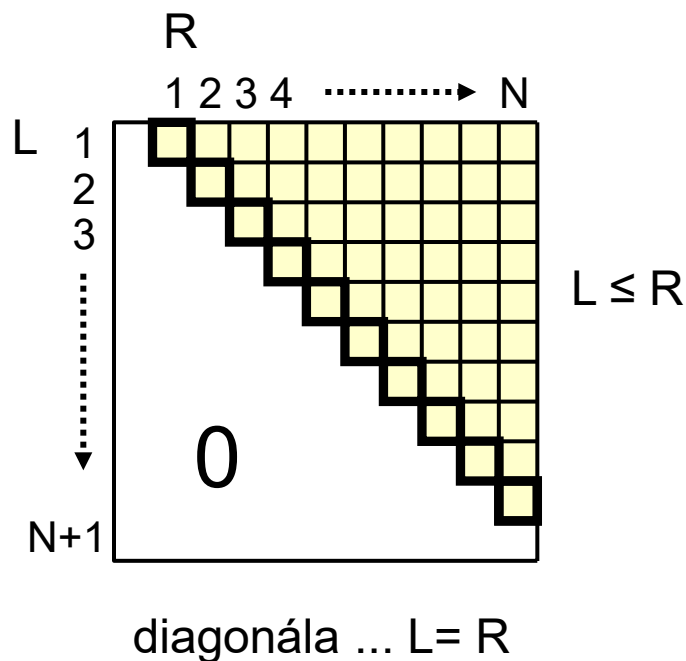
$$\begin{aligned} C(L,R) &= \min_{L \leq k \leq R} \left\{ C(L, k-1) + \sum_{i=L}^{k-1} p_i + C(k+1,R) + \sum_{i=k+1}^R p_i + p_k \right\} = \\ &= \min_{L \leq k \leq R} \left\{ C(L, k-1) + C(k+1,R) + \sum_{i=L}^R p_i \right\} = \\ (*) &= \min_{L \leq k \leq R} \left\{ C(L, k-1) + C(k+1,R) \right\} + \sum_{i=L}^R p_i \end{aligned}$$

Hodnota k minimalizující (*) je indexem kořenu optim. podstromu.

Datové struktury pro výpočet optimálního BVS

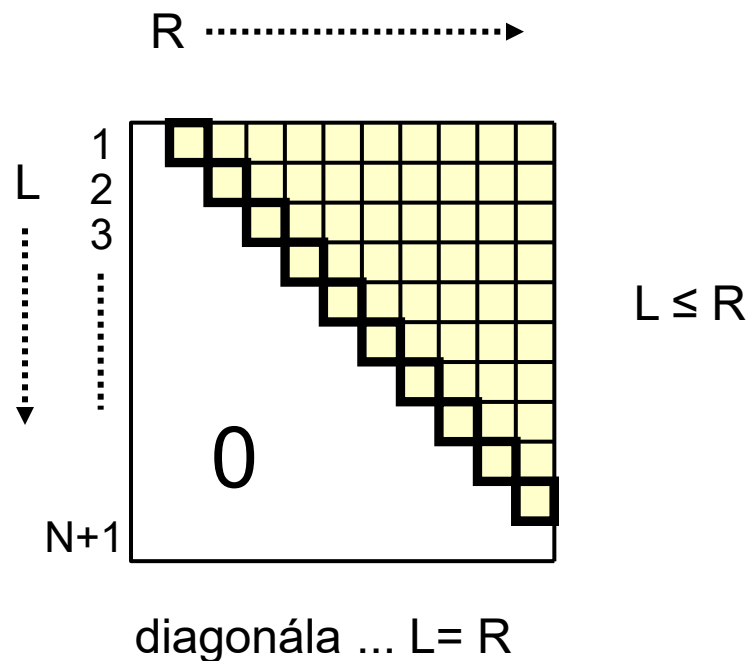
Ceny optimálních podstromů

pole $C[L][R]$ ($L \leq R$)



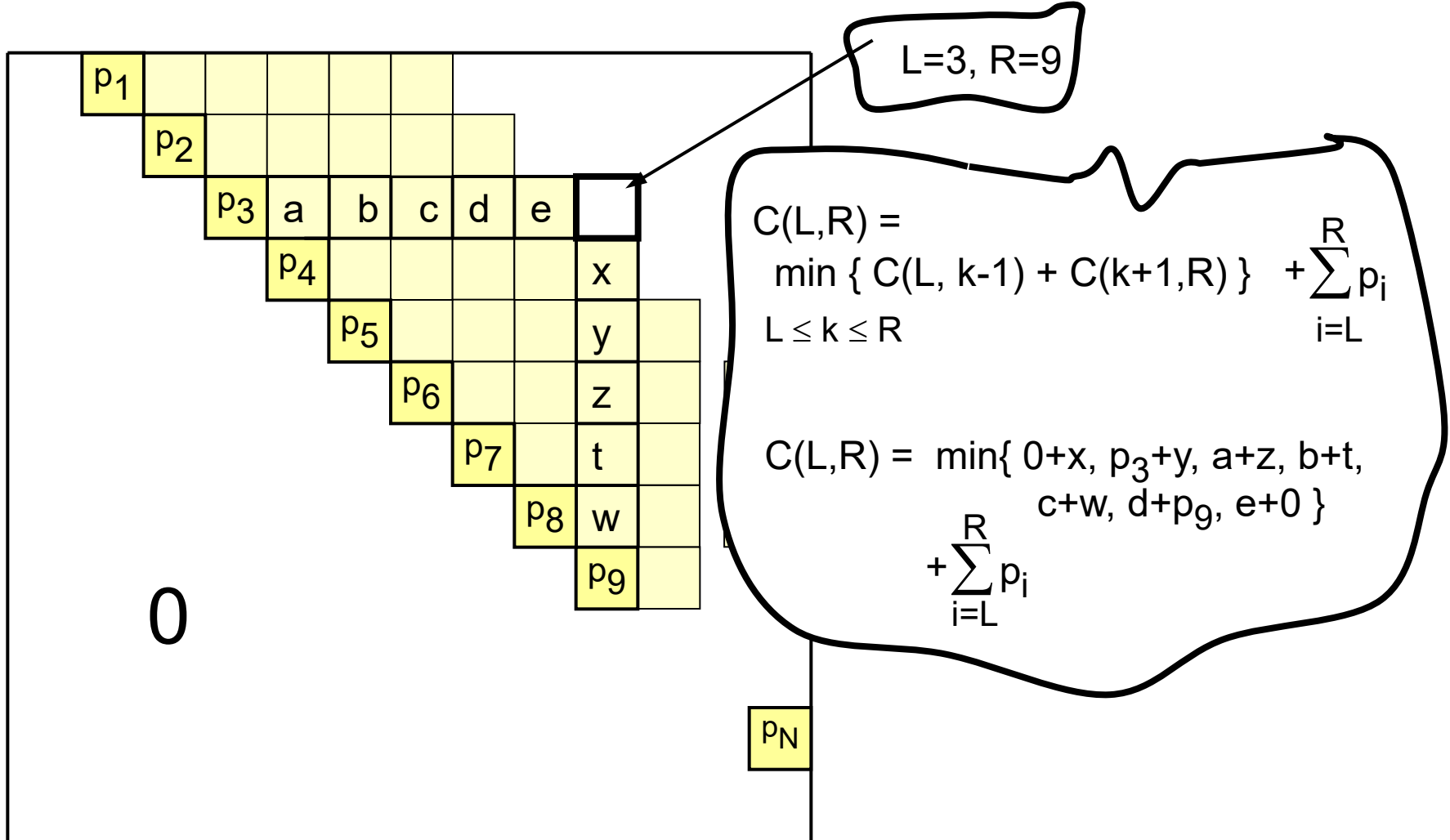
Kořeny optimálních podstromů

pole $roots[L][R]$ ($L \leq R$)



Výpočet optimálního BVS

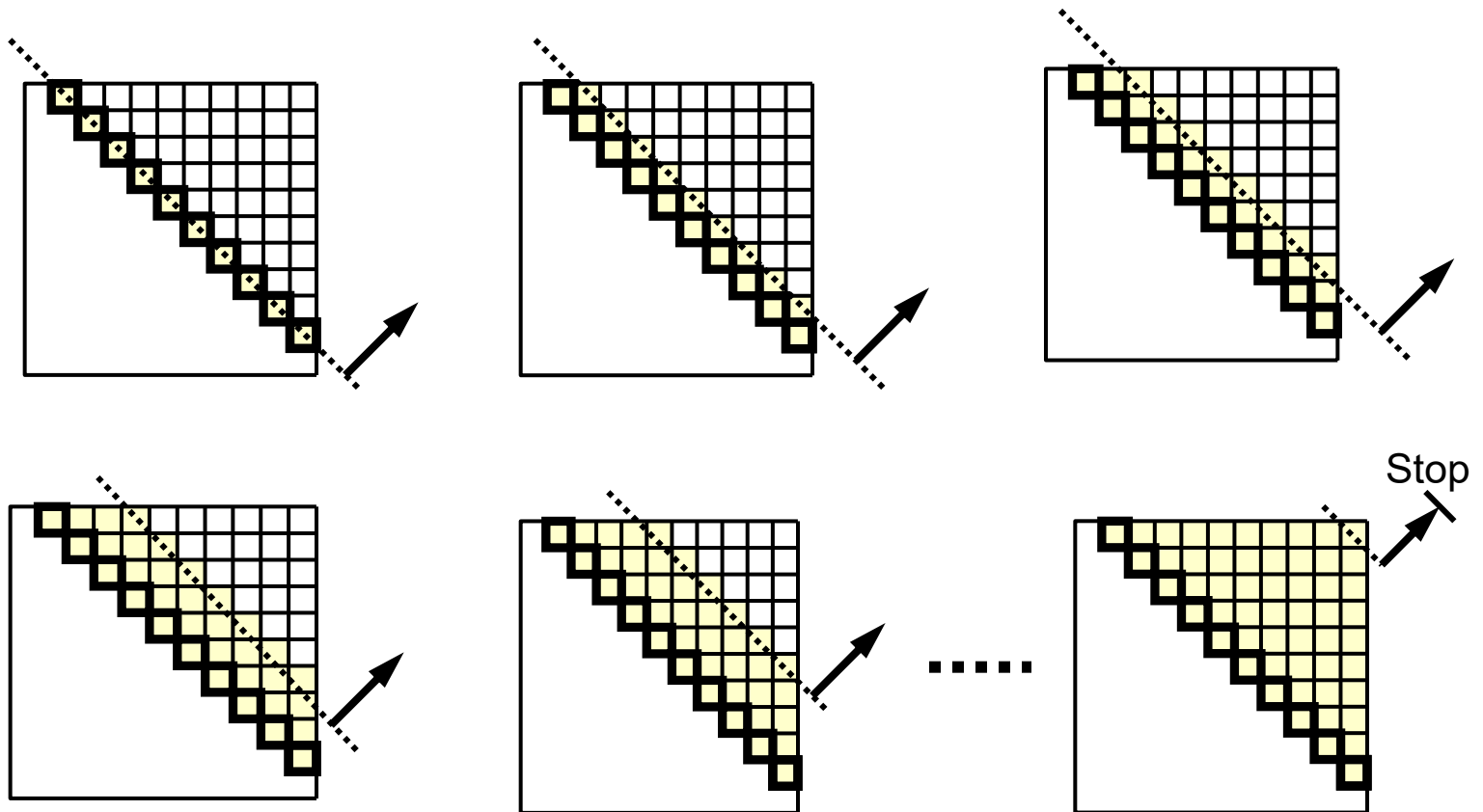
Cena konkrétního optimálního podstromu



Výpočet optimálního BVS

Strategie DP

- nejprve se zpracují nejmenší podstromy, pak větší, atd...



Výpočet optimálního BVS

Výpočet DP tabulek cen a kořenů

```
void optimalTree() {  
    int L, R; double min;  
  
    // size = 1  
    for( i=0; i<=N; i++ ) {  
        C[i][i] = pravděpodobnost[i]; roots[i][i] = i;  
  
        // size > 1  
        for( int size = 2; size <= N; size++ ) {  
            L = 1; R = size;  
            while( R <= N ) {  
                C[L][R] = min(C[L][k-1]+C[k+1][R], k = L..R);  
                roots[L][R] = 'k minimalizující předch. řádek';  
                C[L][R] += sum(C[i][i], i = L..R);  
                L++; R++;  
            }  
        }  
    }
```

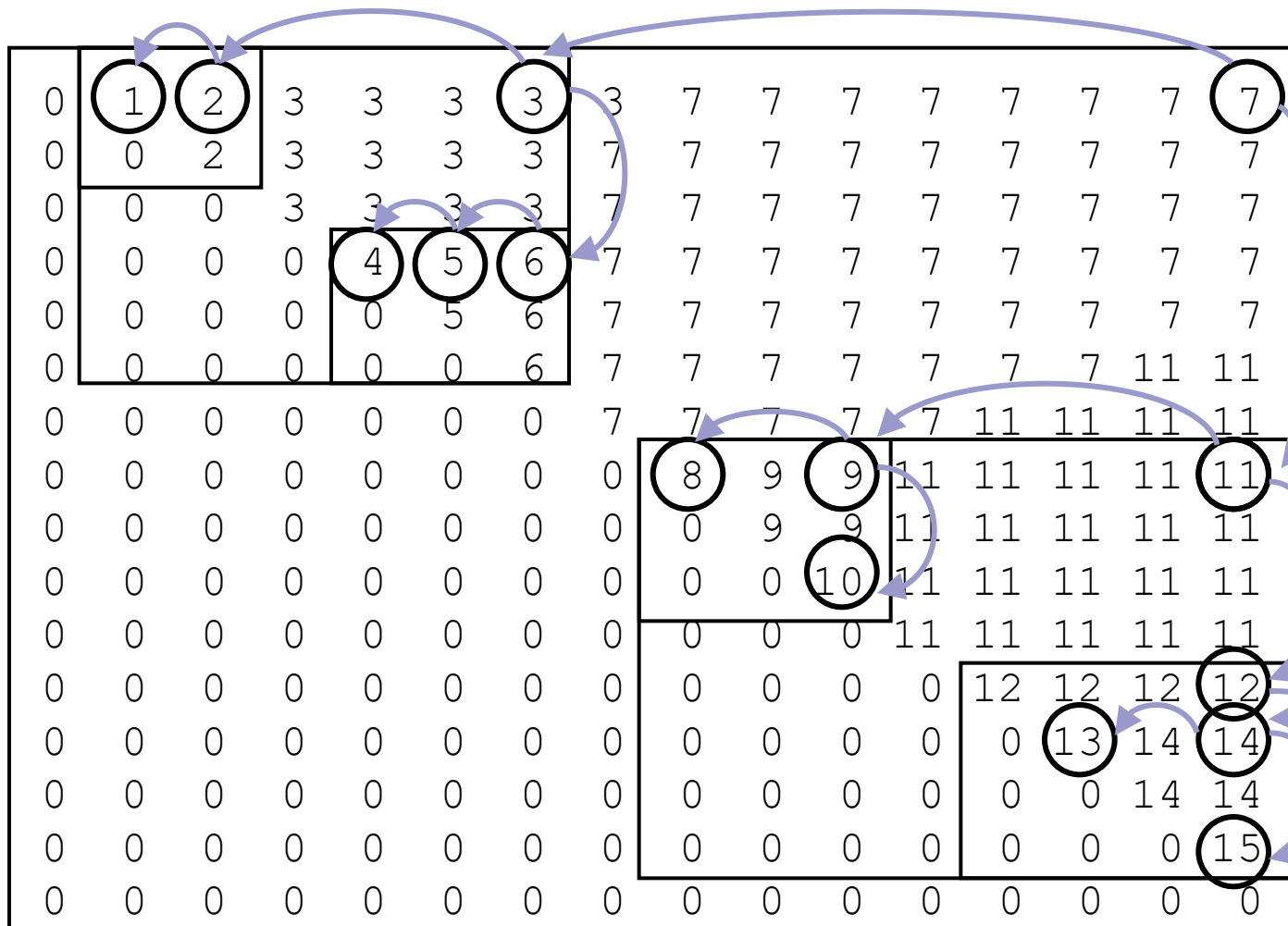
Výpočet optimálního BVS

Vybudování optimálního stromu pomocí
rekonstrukční tabulky kořenů

```
void buildTree( int L, int R) {  
  
    if (R < L) return;  
  
    int keyIndex = roots[L][R];  
    // keys ... sorted array of keys  
    int key = keys[roots[L][R]];  
  
    insert(root, key);    // standard BST insert  
    buildTree( L, keyIndex -1 );  
    buildTree( keyIndex +1, R );  
}
```

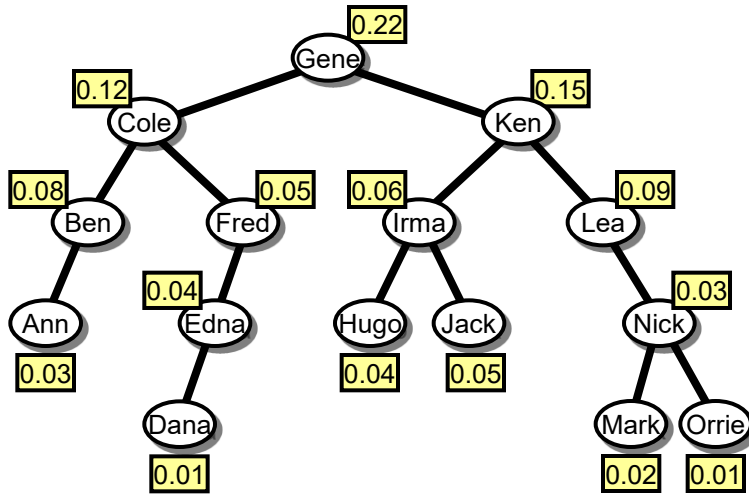
Výpočet optimálního BVS

Kořeny optimálních podstromů

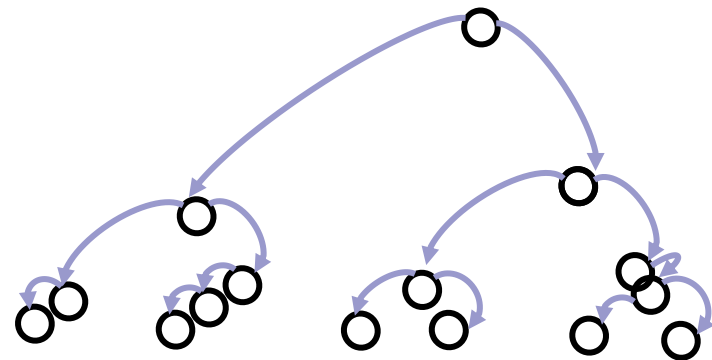
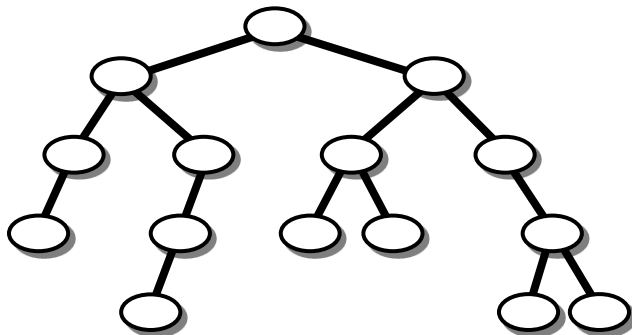


Výpočet optimálního BVS

Korespondence stromů



0	1	2	3	3	3	3	3	7	7	7	7	7	7	7	7	7
0	0	2	3	3	3	3	7	7	7	7	7	7	7	7	7	7
0	0	0	3	3	3	3	7	7	7	7	7	7	7	7	7	7
0	0	0	0	4	5	6	7	7	7	7	7	7	7	7	7	7
0	0	0	0	0	5	6	7	7	7	7	7	7	7	7	7	7
0	0	0	0	0	0	6	7	7	7	7	7	7	7	11	11	11
0	0	0	0	0	0	0	7	7	7	7	7	11	11	11	11	11
0	0	0	0	0	0	0	0	8	9	9	11	11	11	11	11	11
0	0	0	0	0	0	0	0	0	9	9	11	11	11	11	11	11
0	0	0	0	0	0	0	0	0	0	10	11	11	11	11	11	11
0	0	0	0	0	0	0	0	0	0	0	11	11	11	11	11	11
0	0	0	0	0	0	0	0	0	0	0	0	12	12	12	12	12
0	0	0	0	0	0	0	0	0	0	0	0	0	13	14	14	14
0	0	0	0	0	0	0	0	0	0	0	0	0	0	14	14	14
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	15	15
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



Výpočet optimálního BVS

Ceny optimálních podstromů

	1-A	2-B	3-C	4-D	5-E	6-F	7-G	8-H	9-I	10-J	11-K	12-L	13-M	14-N	15-O
1-A	0.03	0.14	0.37	0.39	0.48	0.63	1.17	1.26	1.42	1.57	2.02	2.29	2.37	2.51	2.56
2-B	0	0.08	0.28	0.30	0.39	0.54	1.06	1.14	1.30	1.45	1.90	2.17	2.25	2.39	2.44
3-C	0	0	0.12	0.14	0.23	0.38	0.82	0.90	1.06	1.21	1.66	1.93	2.01	2.15	2.20
4-D	0	0	0	0.01	0.06	0.16	0.48	0.56	0.72	0.87	1.32	1.59	1.67	1.81	1.86
5-E	0	0	0	0	0.04	0.13	0.44	0.52	0.68	0.83	1.28	1.55	1.63	1.77	1.82
6-F	0	0	0	0	0	0.05	0.32	0.40	0.56	0.71	1.16	1.43	1.51	1.63	1.67
7-G	0	0	0	0	0	0	0.22	0.30	0.46	0.61	1.06	1.31	1.37	1.48	1.52
8-H	0	0	0	0	0	0	0	0.04	0.14	0.24	0.54	0.72	0.78	0.89	0.93
9-I	0	0	0	0	0	0	0	0	0.06	0.16	0.42	0.60	0.66	0.77	0.81
10-J	0	0	0	0	0	0	0	0	0	0.05	0.25	0.43	0.49	0.60	0.64
11-K	0	0	0	0	0	0	0	0	0	0	0.15	0.33	0.39	0.50	0.54
12-L	0	0	0	0	0	0	0	0	0	0	0	0.09	0.13	0.21	0.24
13-M	0	0	0	0	0	0	0	0	0	0	0	0	0.02	0.07	0.09
14-N	0	0	0	0	0	0	0	0	0	0	0	0	0	0.03	0.05
15-O	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.01

Audience Q&A



① The Slido app must be installed on every computer you're presenting from

Úloha batohu neomezená

Úloha batohu / Knapsack problem

Máme N předmětů, každý s váhou V_i a cenou C_i ($i = 1, 2, \dots, N$) a batoh s kapacitou váhy K .

Máme naložit batoh těmito předměty tak, aby kapacita K nebyla překročena a obsah měl maximální cenu.

Neomezená varianta -- Každý předmět lze použít libovolněkrát.

0/1 varianta -- Každý předmět lze použít nejvýše jednou.

Schematický batoh s kapacitou 10



30

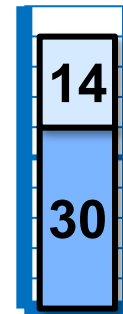
16

14

9

Předměty
s uvedenou
cenou,
váha ~ výška

Několik možných konfigurací



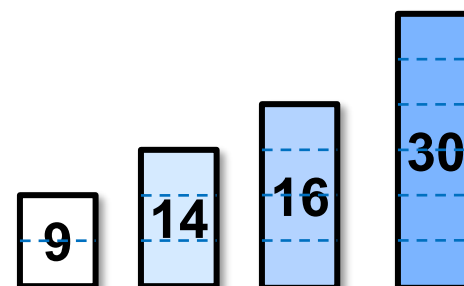
Úloha batohu neomezená

Příklad

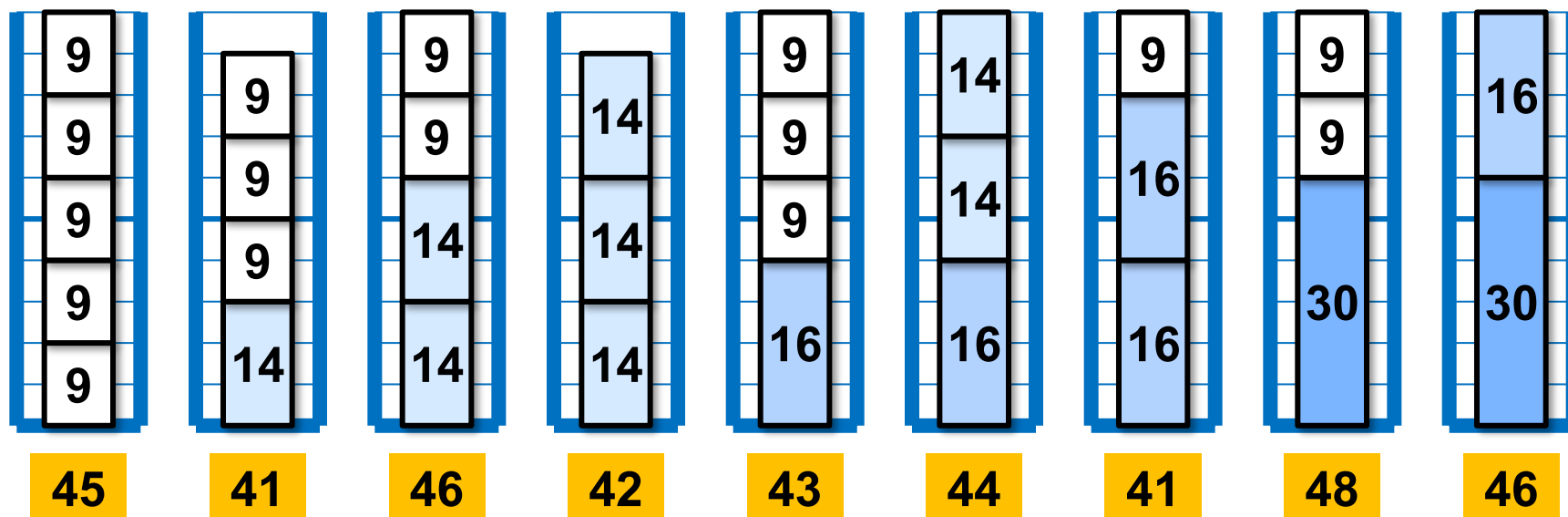
$N = 4$

Váha	2	3	4	6
Cena	9	14	16	30

Batoh s kapacitou 10



Některé možnosti naplnění a odpovídající ceny



Jak bude úloha neomezeného batohu řešena?



① The Slido app must be installed on every computer you're presenting from

Úloha batohu neomezená

Použijeme $K+1$ batohů, o kapacitách $0, 1, 2, 3, \dots, K$.

Hodnotu optimálního naplnění batohu s kapacitou K lze získat jako maximum z hodnot

- (optimální naplnění batohu o kapacitě $K - V_1$) + C_1 ,
- (optimální naplnění batohu o kapacitě $K - V_2$) + C_2 ,
- ...
- (optimální naplnění batohu o kapacitě $K - V_N$) + C_N .

Optimální naplnění batohu o kapacitě $K - V_i$ ($i = 1..N$) je stejnou úlohou, jen s menšími daty. Hodnoty předpočítáme standardně metodou DP do 1D tabulky.

Neomezenou úlohu batohu lze přímo vyjádřit jako úlohu nalezení nejdelší cesty v DAG. Postup řešení je identický.

Převod na DAG

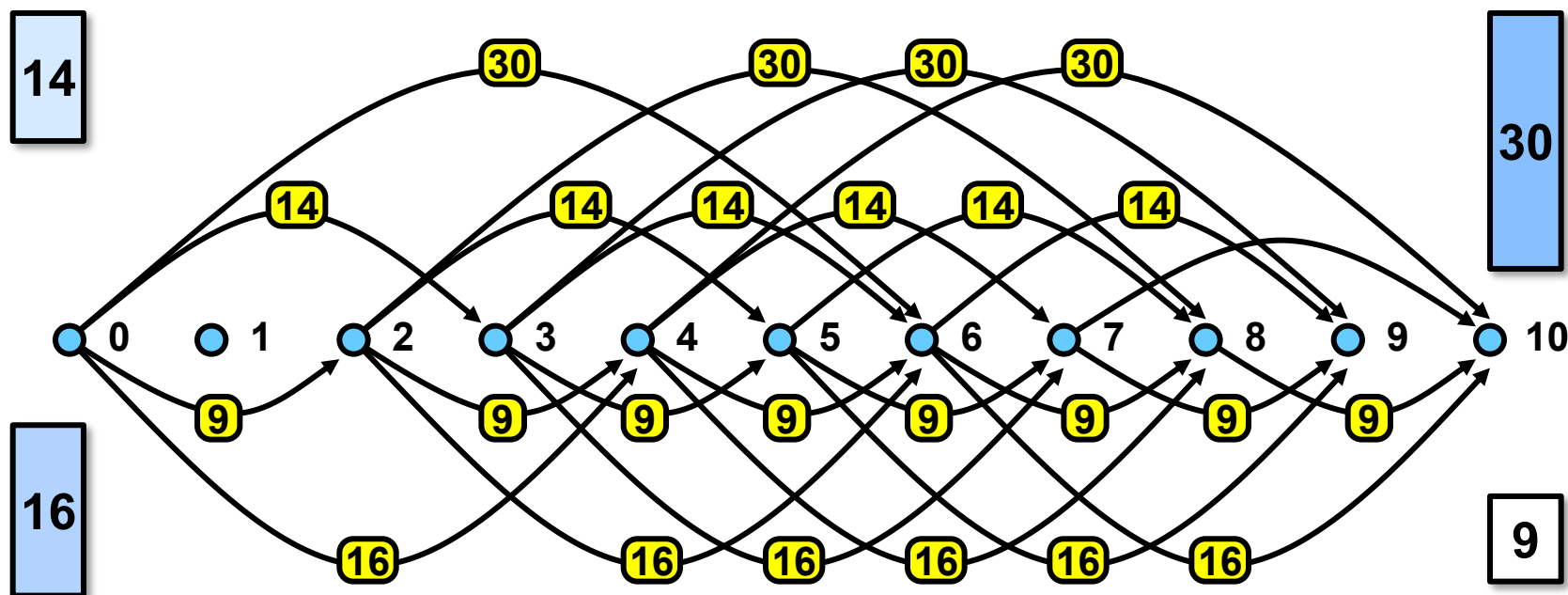
DAG:

Uzly: Kapacity 0, 1, 2, 3, ..., K.

Hrany: Z uzlu X vedou hrany po řadě do uzlů $X+V_1$, $X+V_2$, ..., $X+V_N$, jsou po řadě ohodnoceny cenami C_1 , C_2 , ..., C_N .

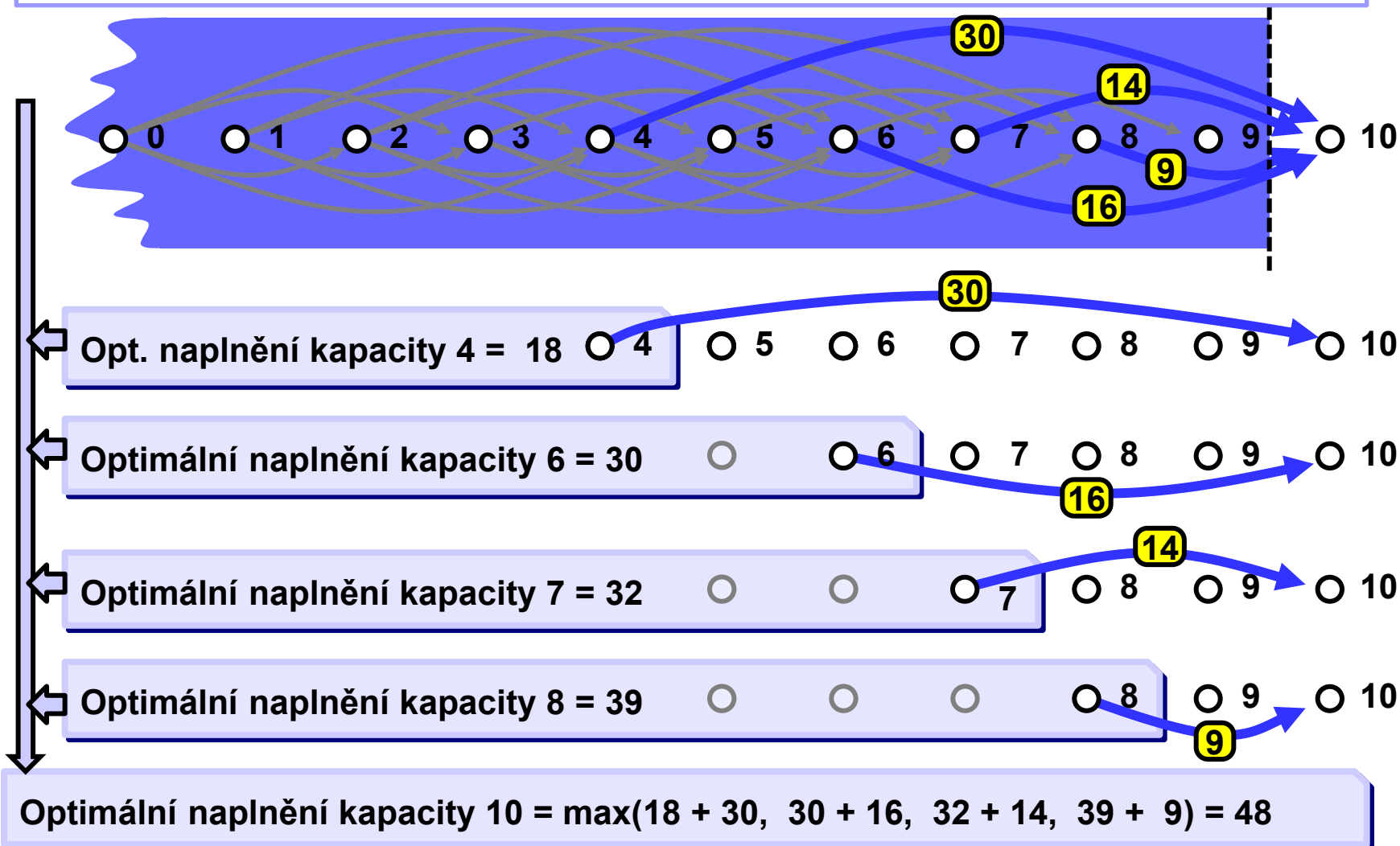
Příklad

$K = 10$, $N = 4$, $V_i = (2, 3, 4, 6)$, $C_i = (9, 14, 16, 30)$, $i = 1..4$.



Neomezená úloha batohu -- jako DAG

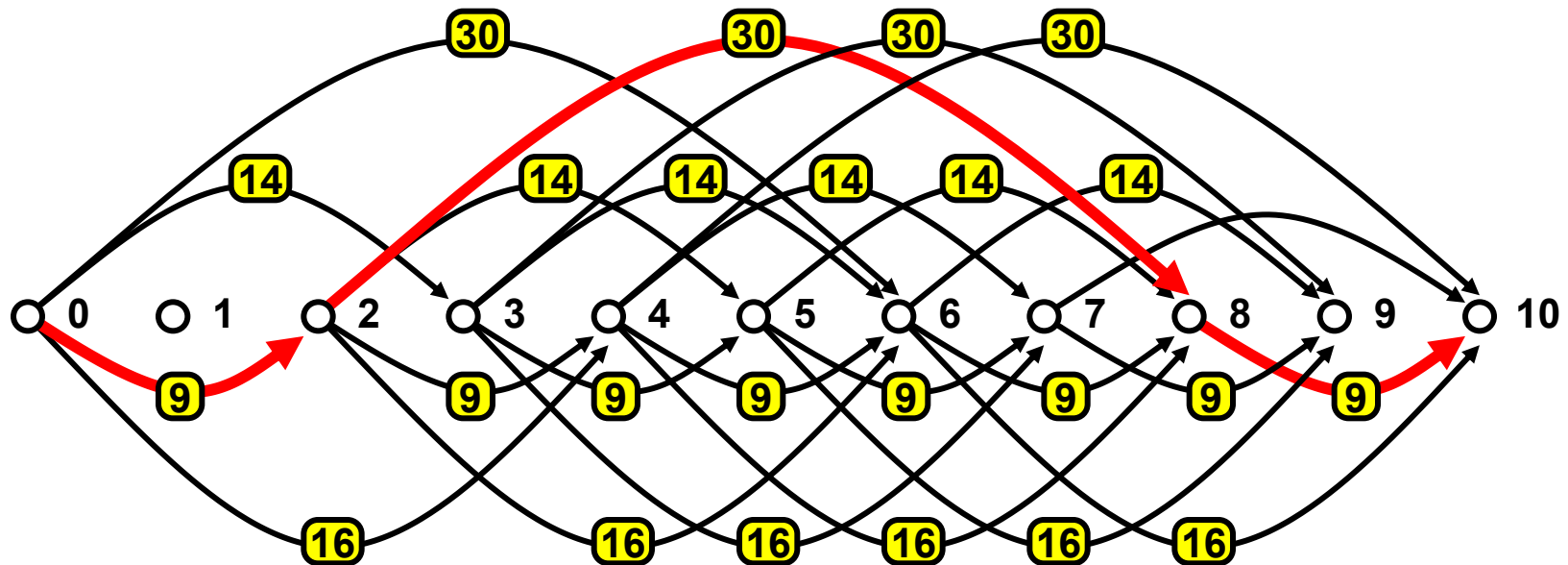
Optimální naplnění kapacity 10 = ??



Úloha batohu neomezená

Nejdelší cesta odpovídá optimálnímu naplnění batohu.
Dvě hrany s cenou 9 a jedna hrana s cenou 30, celkem cena = 48.

Batoh optimálně naplníme dvěma předměty s váhou 2 a cenou 9
a jedním předmětem s váhou 6 a cenou 30.

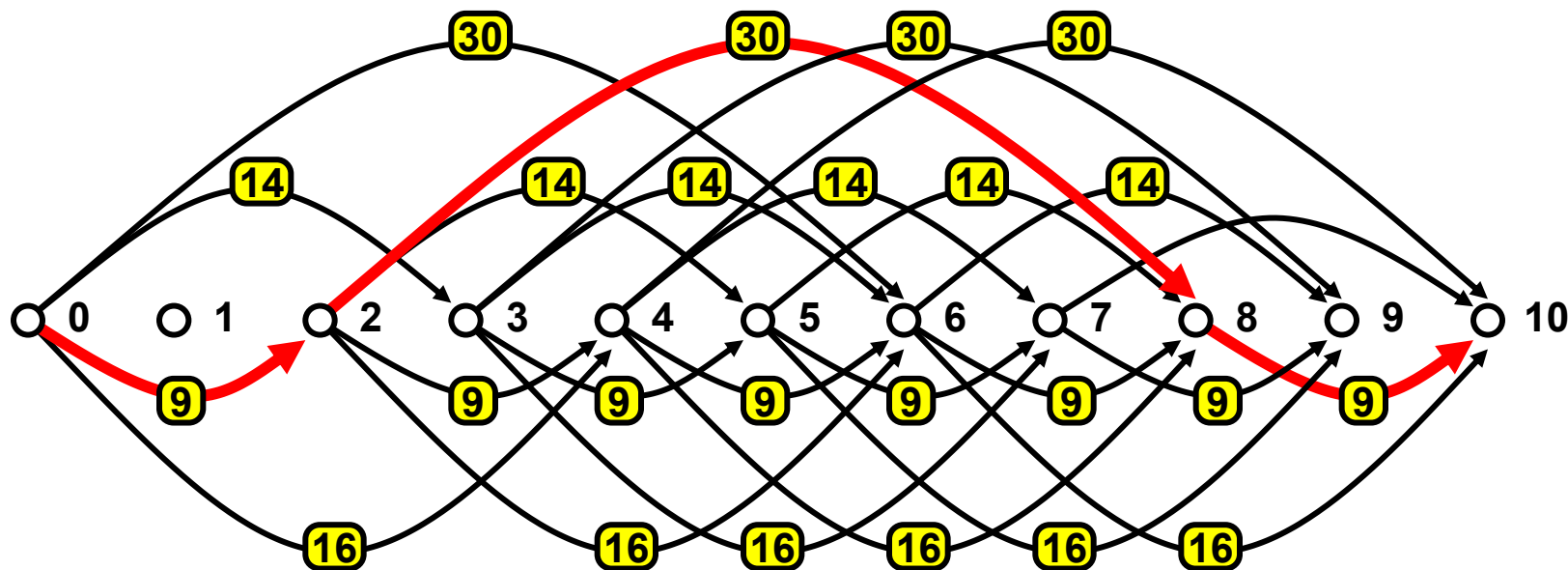


Úloha batohu neomezená - asymptotická složitost

DAG obsahuje $K+1$ uzlů a méně než $K*N$ hran.

Má tedy $V = \Theta(K)$ uzlů a $E = O(K*N)$ hran.

Asymptotická složitost hledání nejdelší cesty je $\Theta(V+E)$,
máme tedy pro neomezenou úlohu batohu
asymptotickou složitost $O(K + K*N) = O(K*N)$.



Úloha batohu neomezená - asymptotická složitost

Zdánlivá nesrovnalost

1. Literatura: NP těžký problém, není znám efektivní algoritmus.
2. ALG Ol: DP řeší úlohu v čase v $O(N \cdot K)$, tedy efektivně?

Délka výpočtu DP je lineárně závislá na velikosti kapacity K.

Příklad

Velkou kapacitu 2^{64} lze zadat velmi krátkým zápisem

Kapacita = 18446744073709551616.

$N = 3$. Položky (váha, cena): (2, 345), (3, 456), (5, 678).

Vstupní data lze zapsat do cca 100 bitů < 16 Bytů $< \text{"dva longy"}$

Výpočet pomocí DP potrvá přes 584 roky

za předpokladu, že za 1 sec vyplní 10^9 prvků tabulky.

Délka výpočtu DP je exponenciálně závislá na délce řetězce definujícího kapacitu K.

Audience Q&A



① The Slido app must be installed on every computer you're presenting from

0/1 úloha batohu

- Každý předmět lze použít nejvýše 1 krát.

Máme vybrat vhodnou podmnožinu předmětů splňující zadání úlohy. Každé podmnožině lze přiřadit charakteristický vektor z hodnot 0/1 délky N . Pozice ve vektoru odpovídá předmětu, 0 resp. 1 odpovídá nepřítomnosti resp. přítomnosti předmětu v této podmnožině. Binárních vektorů délky N je celkem 2^N , systematické probírání všech možných podmnožin bude mít exponenciální asymptotickou složitost, nehodí se.

- DP poskytuje (pro relativně nevelké kapacity) výhodnější postup.

0/1 úloha batohu

Příklad

N = 4

Váha Cena

2 9

3 14

4 16

6 30

9

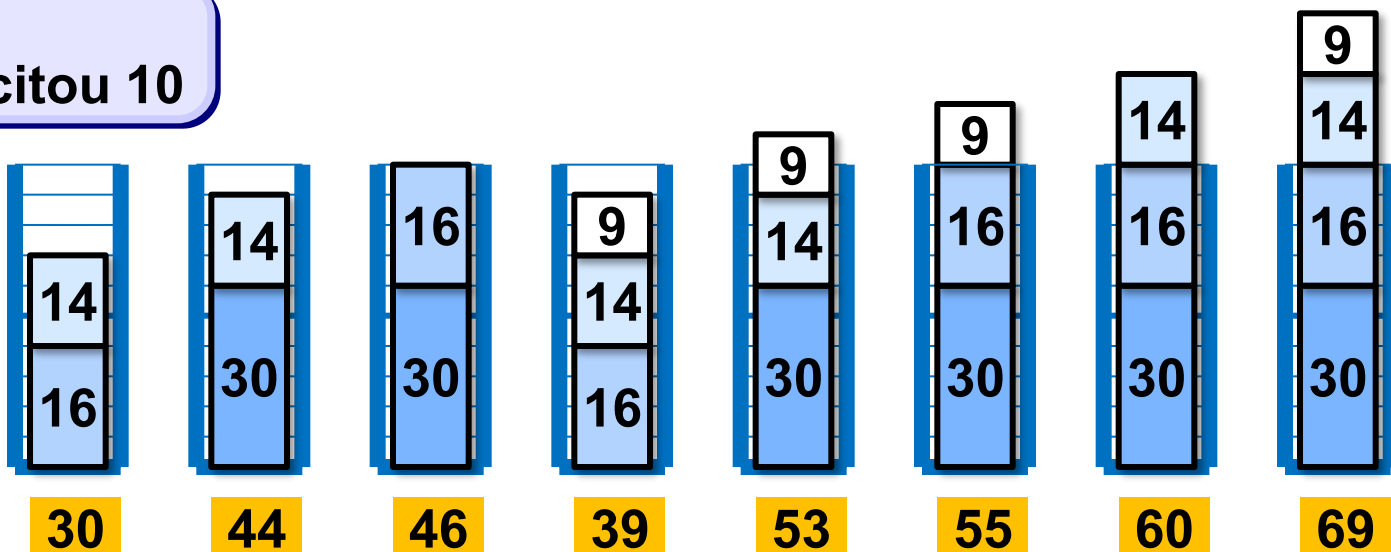
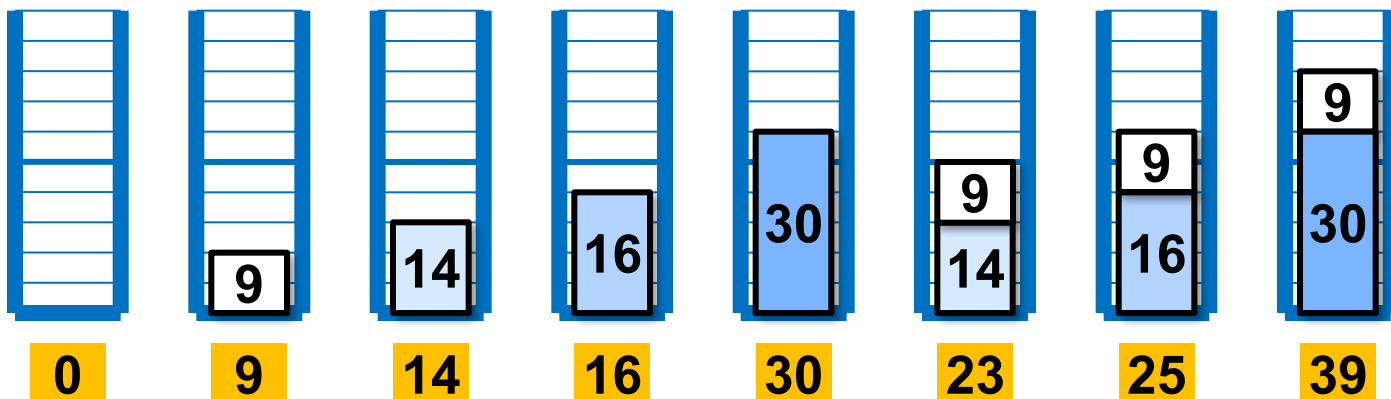
14

16

30

Batoh
s kapacitou 10

Všech 16 podmnožin čtyř předmětů a jejich ceny



0/1 úloha batohu - řešení

Použijeme $K+1$ batohů, o kapacitách $0, 1, 2, 3, \dots, K$.
Použijeme $N+1$ souborů předmětů.

Soubor 0 neobsahuje žádný předmět.

Soubor 1 obsahuje předmět 1.

Soubor 2 obsahuje předměty 1 a 2.

Soubor 3 obsahuje předměty 1, 2, 3.

...

Soubor N obsahuje předměty $1, 2, 3, \dots, N$.

Na pořadí předmětů nezáleží, je ale zafixované.

Pro každou kapacitu a pro každý soubor budeme řešit stejnou úlohu metodou DP, v pořadí od menších hodnot k větším.

0/1 úloha batohu - řešení

Označme symbolem $U(x, y)$ úlohu se souborem předmětů $1, 2, \dots, x$ a s kapacitou batohu y a symbolem $Opt(x, y)$ optimální řešení této úlohy.

Pro řešení $U(x, y)$ použijeme optimální řešení úloh $U(x-1, _)$:

- Bud' do $Opt(x, y)$ zahrneme předmět x nebo jej nezahrneme. V prvním případě použijeme hodnotu řešení pro batoh s kapacitou menší o velikost váhy V_x , tedy hodnotu $Opt(x-1, y-V_x)$, ke které přičteme cenu C_x předmětu x .
- V druhém případě beze změny použijeme hodnotu $Opt(x-1, y)$. Z obou hodnot vybereme tu výhodnější a dostáváme tak:

$$Opt(x, y) = \max(Opt(x-1, y), Opt(x-1, y-V_x) + C_x).$$

Dále zřejmě platí $Opt(0, y) = Opt(x, 0) = 0$, pro $x = 0..N$, $y = 0..K$.

0/1 úloha batohu - řešení

Pro $x = 1..N$, $y = 0..K$:

- $\text{Opt}(x, y) = \max(\text{Opt}(x-1, y), \text{Opt}(x-1, y-Vx) + Cx)$.
- $\text{Opt}(0, y) = \text{Opt}(x, 0) = \text{Opt}(0, 0) = 0$.

Pokud $y-Vx < 0$, položíme $\text{Opt}(x, y-Vx) = -\infty$ (a netabelujeme).

Hodnoty $\text{Opt}(x,y)$ tabelujeme ve 2D tabulce velikosti $(N+1) \times (K+1)$ s řádkovým indexem x (předměty) a sloupcovým indexem y (kapacity menších batohů).

Pro rekonstrukci optimálního řešení použijeme tabulku předchůdců stejné velikosti jako tabulku pro $\text{Opt}(x, y)$. Předchůdce leží vždy v předchozím řádku $x-1$, stačí registrovat buď pozici y (beze změny) nebo pozici $y-Vx$ (přidán předmět x).

0/1 úloha batohu

Příklad

N = 4

Kapacita = 10

Váha	2	3	4	6
Cena	9	14	16	30

9

14

16

30



Opt(x, y)

	0	1	2	3	4	5	6	7	8	9	10
0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	9	9	9	9	9	9	9	9	9
2	0	0	9	14	16	23	23	23	23	23	23
3	0	0	9	14	16	23	25	30	30	39	39
4	0	0	9	14	16	23	30	30	39	44	46

Pred(x, y)

	0	1	2	3	4	5	6	7	8	9	10
0	--	--	--	--	--	--	--	--	--	--	--
1	0	1	0	1	2	3	4	5	6	7	8
2	0	1	2	0	1	2	3	4	5	6	7
3	0	1	2	3	0	5	2	3	4	5	6
4	0	1	2	3	4	5	0	7	2	3	4

0/1 úloha batohu

Vyjádření jako optimální cesty v DAG

Uzly DAG budou jednotlivé hodnoty $\text{Opt}(x, y)$, $x = 0..N$, $y = 0..K$, celkem bude mít DAG $(N+1)*(K+1)$ uzlů.

Do uzlu $\text{Opt}(x, y)$ povede hrana

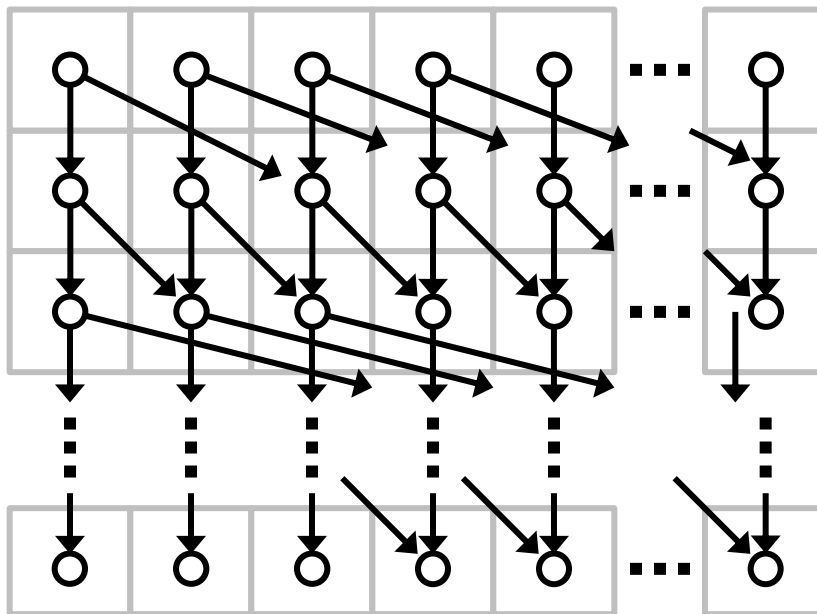
- $\text{Opt}(x-1, y) \rightarrow \text{Opt}(x, y)$
ohodnocená 0 (žádný přidaný předmět),
- a pokud $y - Vx \geq 0$, také hrana
 $\text{Opt}(x-1, y - Vx) \rightarrow \text{Opt}(x, y)$
ohodnocená cenou Cx (cenou přidaného předmětu x).

V takto zkonstruovaném DAG hledáme nejdelší (= nejčennější) cestu standardní DP metodou.

Jaké je topologické uspořádání tohoto DAG?

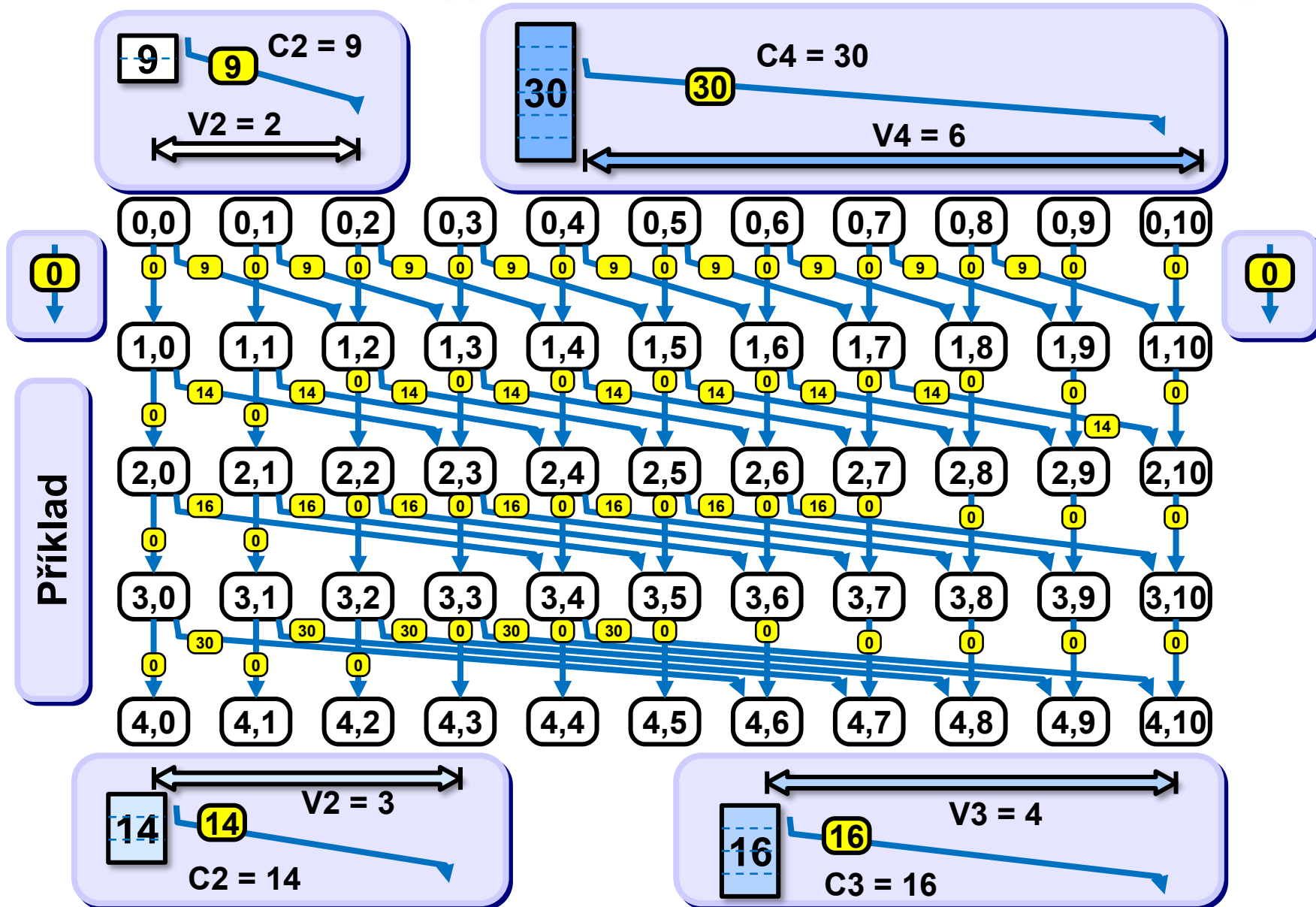
0/1 úloha batohu - topologické uspořádání DAG

DAG můžeme uvažovat nakreslený formálně do DP tabulky, přičemž uzel $\text{Opt}(x, y)$ leží v buňce s indexy x a y . Pak hrany DAG vedou vždy pouze z předchozího řádku do následujícího řádku. Pokud tento DAG procházíme shora po řádcích, to jest ve stejném pořadí, v němž vyplňujeme DP tabulku, respektujeme jeho topologické uspořádání.



V tomto případě není nutno uzly DAG v topologickém uspořádání uvažovat v jedné přímce, "tabulkové" uspořádání je přehlednější.

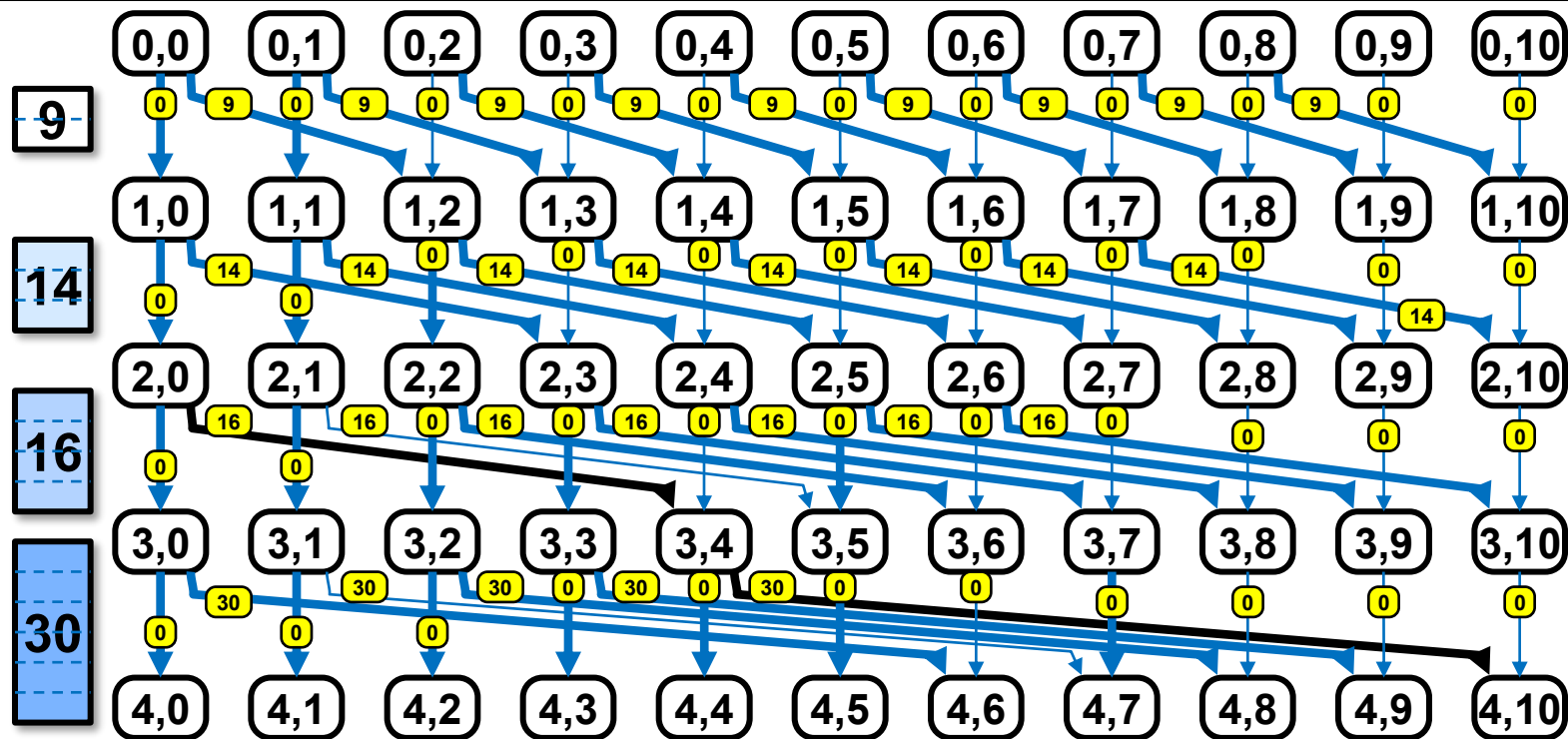
0/1 úloha batoh -- DAG



0/1 úloha batoh -- rekonstrukce optimálního řešení

tabulka předchůdců

Pred(x, y)	0	1	2	3	4	5	6	7	8	9	10
0	--	--	--	--	--	--	--	--	--	--	--
1	0	1	0	1	2	3	4	5	6	7	8
2	0	1	2	0	1	2	3	4	5	6	7
3	0	1	2	3	0	5	2	3	4	5	6
4	0	1	2	3	4	5	0	7	2	3	4



0/1 úloha batoh -- Asymptotická složitost

Tabulka ... Velikost ... $(N+1)*(K+1) \in \Theta(N*K)$
Vyplnění jedné buňky ... $\Theta(1)$
Vyplnění tabulky ... $\Theta(N*K*1) = \Theta(N*K)$.
Rekonstrukce optimálního řešení $\Theta(N)$.
Celkem ... $\Theta(N*K + N) = \Theta(N*K)$.

DAG Uzlů ... $(N+1)*(K+1) \in \Theta(N*K)$.
Hran ... nejvýše $2*(N+1)*(K+1)$, tj $\in O(N*K)$.
Nalezení optimální cesty ... $\Theta(\text{uzlů}+\text{hran}) = \Theta(N*K)$.

Řešení obou variant úlohy batohu, neomezené i 0/1,
má asymptotickou složitost $\Theta(N*K)$.

Přitom zároveň platí:

**Asymptotická složitost DP řešení je exponenciální
vzhledem k délce řetězce definujícího kapacitu K.**

Audience Q&A



① The Slido app must be installed on every computer you're presenting from