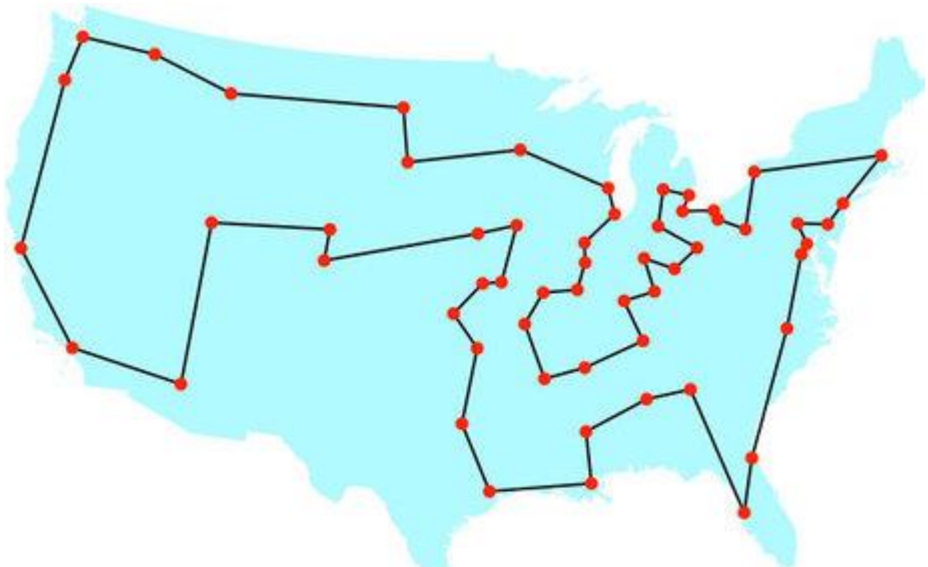
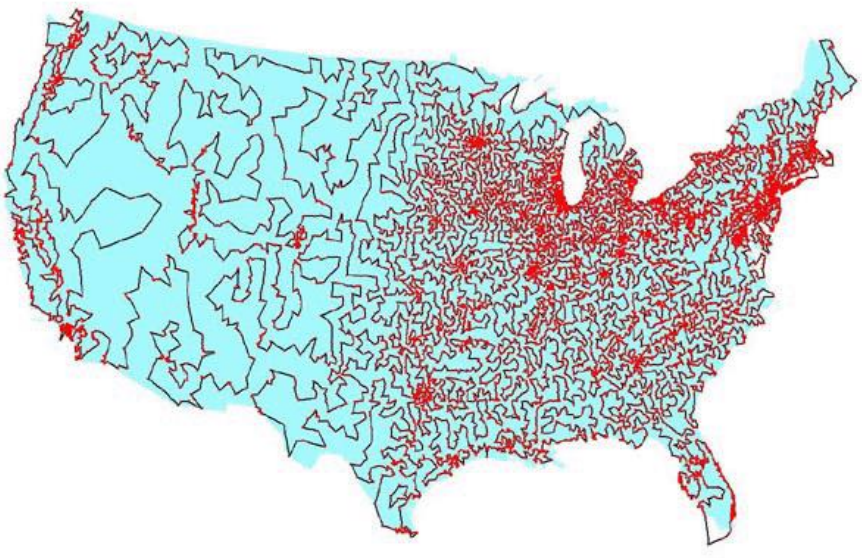
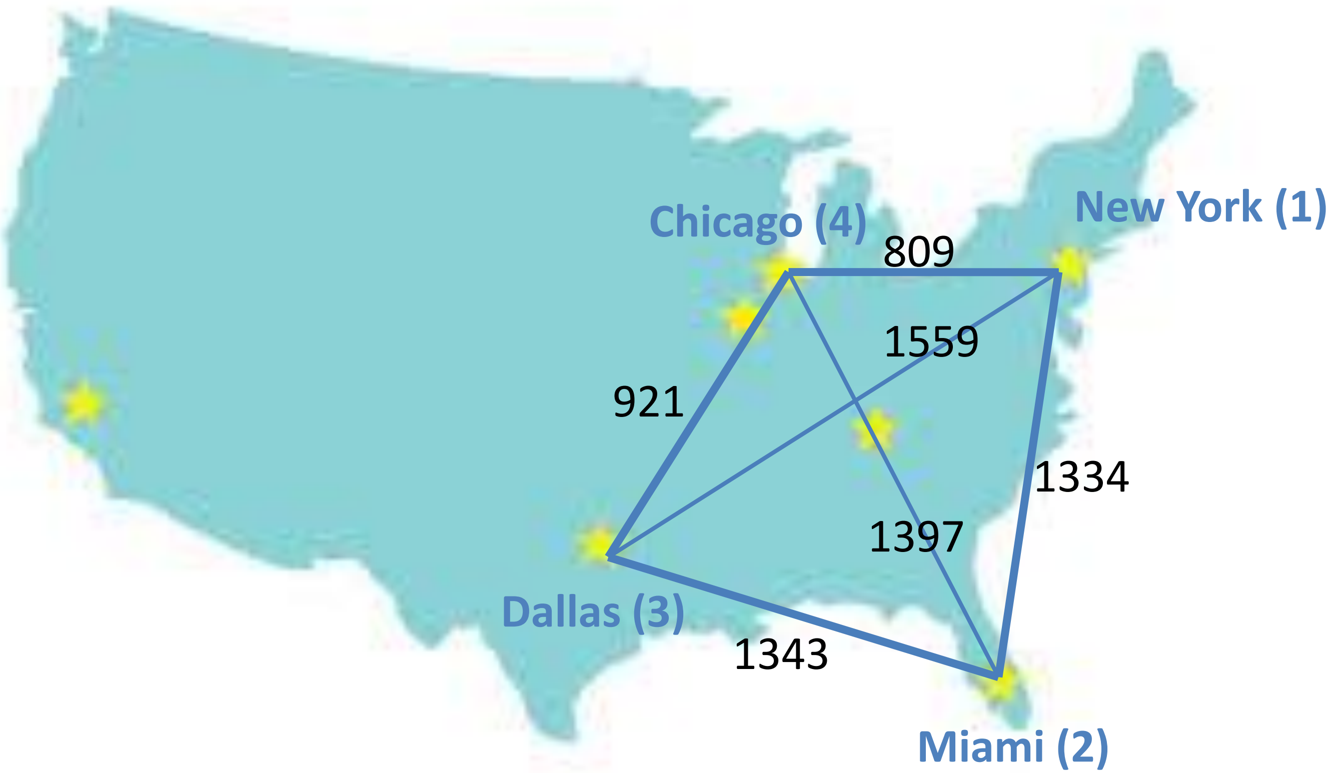
Utazó ügynök probléma (TSP)

A feladat

- Adva van *n* város, illetve az útiköltség bármely két város között, keressük a legolcsóbb utat egy adott városból indulva, amely minden várost pontosan egyszer érint, majd a kiindulási városba ér vissza.

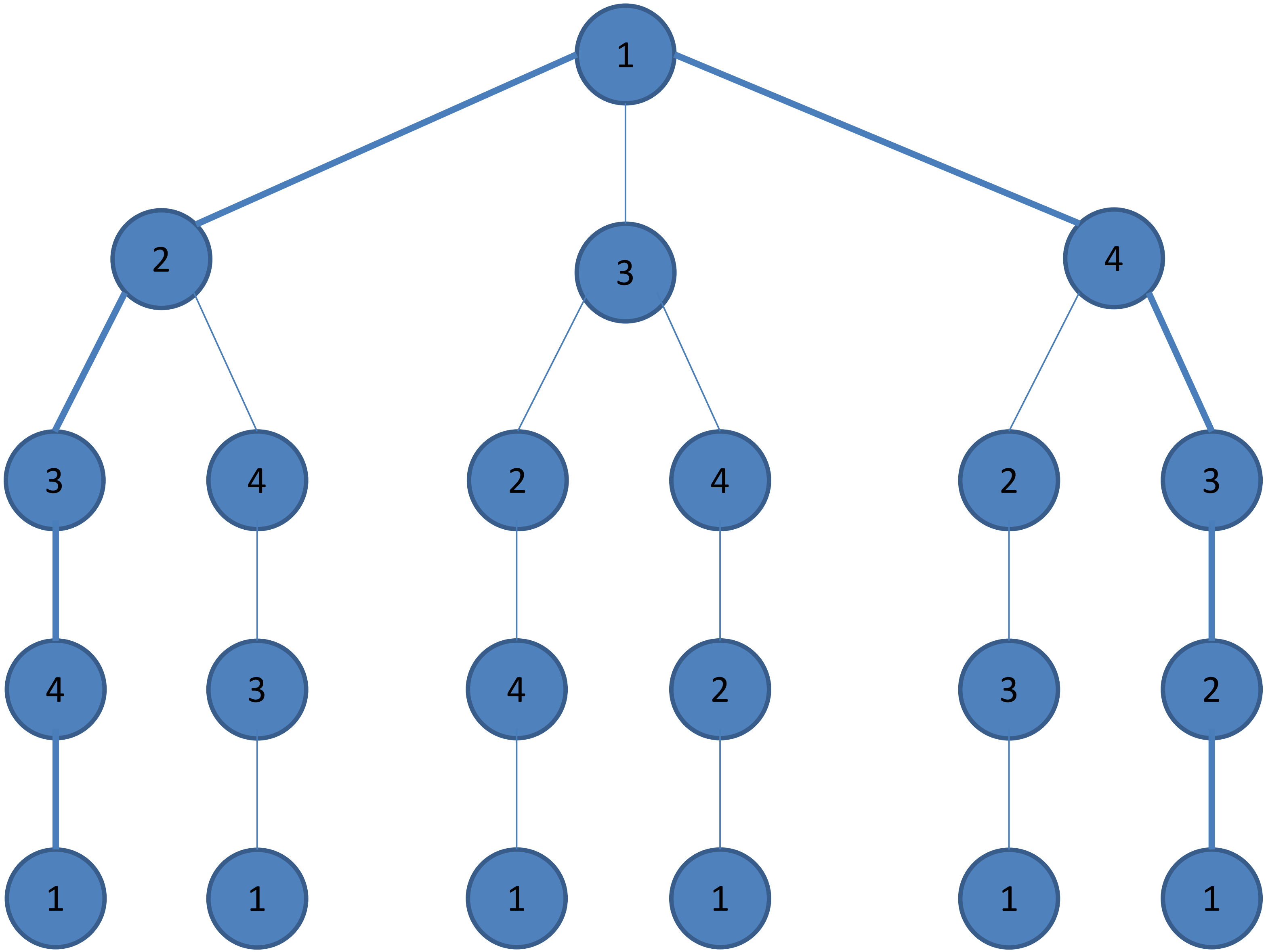






	New York	Miami	Dallas	Chicago
1. New York	—	1334	1559	809
2. Miami	1334	—	1343	1397
3. Dallas	1559	1343	—	921
4. Chicago	809	1397	921	—

Keresési tér



Számítási nehézség

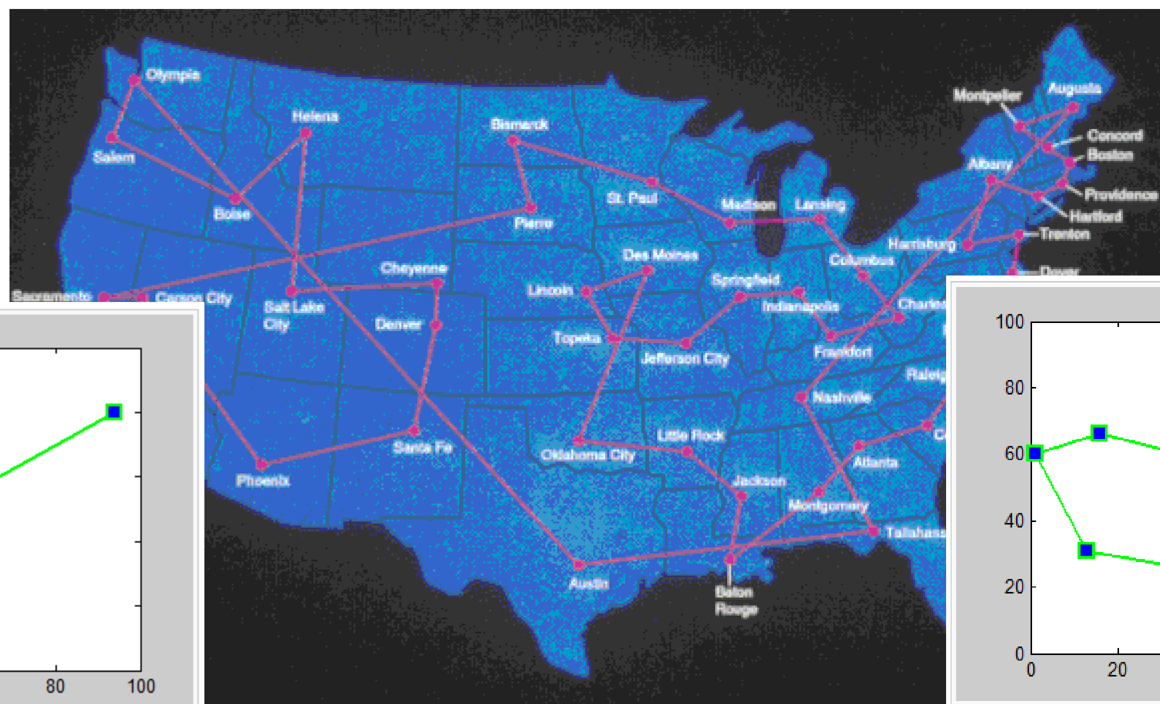
- A legkézenfekvőbb megoldás az összes permutáció végignézése
  - ((n-1)!)/2 út közül kell választanunk egyet, a legrövidebbet
- Dinamikus programozás ($O(n^2 2^n)$)
  - 1962: Bellman / Held–Karp
- **Teljes megoldású algoritmus**
  - 1954: 49 amerikai város
  - 1977: 120 város Németország környékéről
  - 1987: 2392 pont
  - 2004: Svédország 24 978 városa
  - 2006: 85 900 (CONCORDE algoritmus)
    - branch-and-bound módszerrel

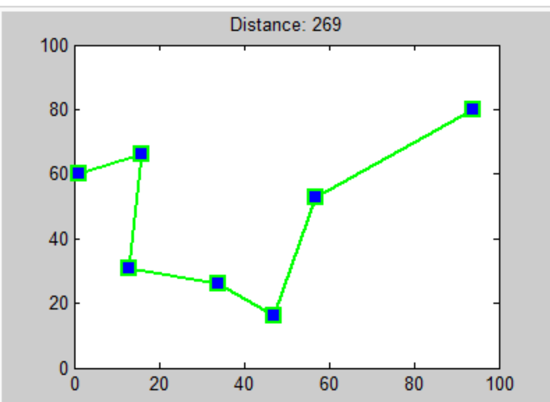
1,2,3,4,1
1,2,4,3,1
1,3,2,4,1
1,3,4,2,1
1,4,2,3,1
1,4,3,2,1

?

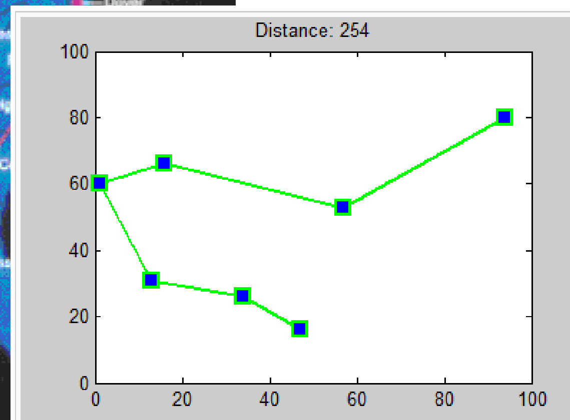
Heurisztikus módszerek (1)

- NN algoritmus (Nearest Neighbor) (GREEDY)
  - Mindig a legközelebbi még nem látogatott csúcsba megyünk





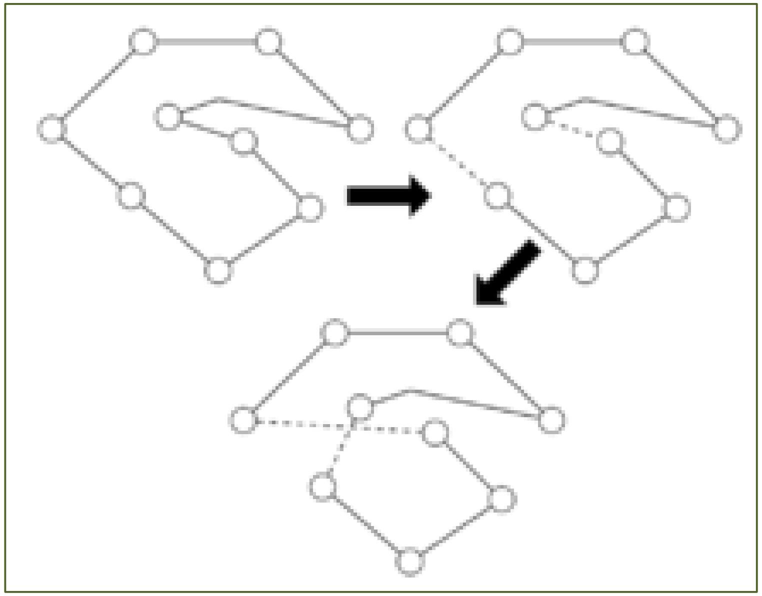
Nearest Neighbor algorithm for a TSP with 7 cities. The solution changes as the starting point is changed



Nearest Neighbor algorithm for a TSP with 7 cities. The solution changes as the starting point is changed

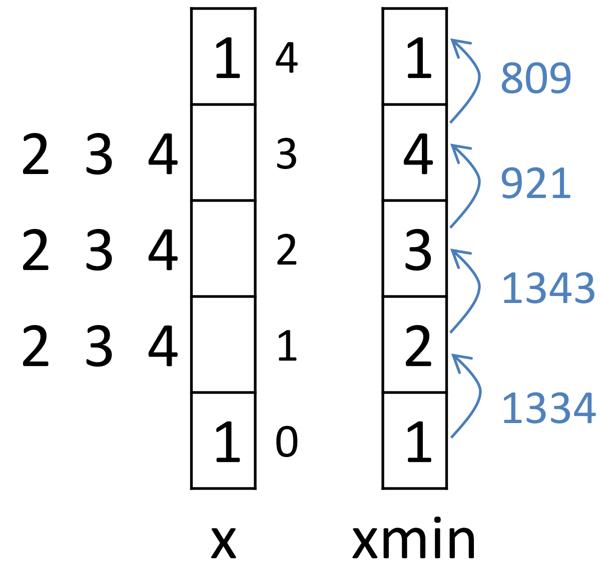
Heurisztikus módszerek (2)

- Véletlenszerűen válassz ki két nem-szomszédos élet a mohó körről, és –amennyiben ez javít a hosszan– cseréld le ezeket arra az él-párra, amely helyreállítja a kört.
  - Ismételd e lépést addig, míg e módszer már eredményez javítást.



Generáljuk a permutációkat (Backtracking)

```
BTp(x[],n,k)
   minden x[k] = 2,n végezd
      ha ígéretes(x,k) akkor
         ha k < n-1 akkor
            BTp(x,n,k+1)
         különben
            frissít_min_ut(xmin,x,n,d)
         vége ha
      vége ha
   vége minden
vége BTp
```




```
ígéretes(x[],k)
   minden i = 1,k-1 végezd
      ha x[i] == x[k] akkor
         return HAMIS
      vége ha
   vége minden
   return IGAZ
vége ígéretes
```

Minimum keresés a generált utak között

d	1	2	3	4
1	0	1334	1559	809
2	1334	0	1343	1397
3	1559	1343	0	921
4	809	1397	921	0

Mohó algoritmus

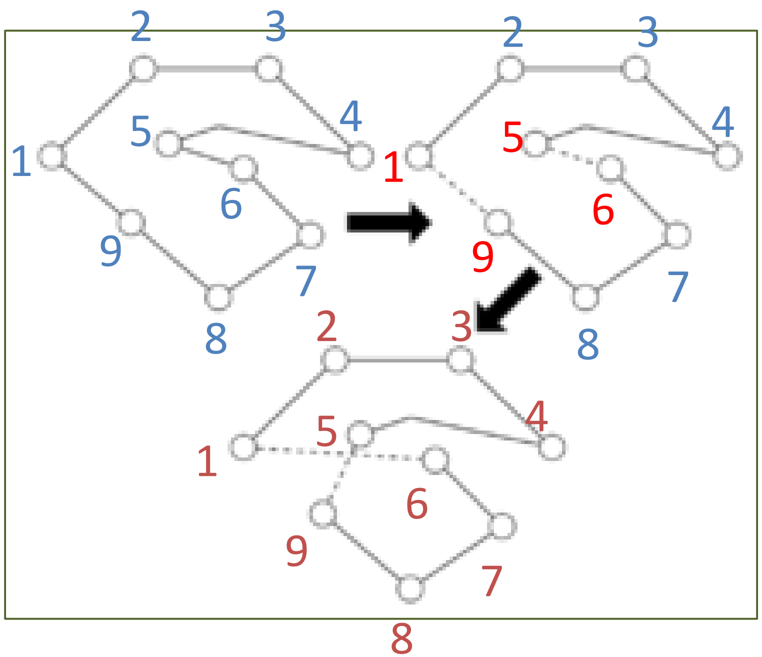
```
int NN(int, int*, int**);

int main(){
    int n,i,*v,*x,**d;
    . . .
    x[0] = x[n] = 1; v[1] = 1;
    for ( i = 1 ; i < n ; ++i ){
        x[i] = NN(x[i-1],v,d); v[x[i]] = 1;
    }…
}
```

d	1	2	3	4
1	0	1334	1559	809
2	1334	0	1343	1397
3	1559	1343	0	921
4	809	1397	921	0

x		v		x		v		x		v		x		v	
1	4			1	4			1	4			1	4		
	3	0	4		3	1	4		3	1	4	2	3	1	4
	2	0	3		2	0	3	3	2	1	3	3	2	1	3
	1	0	2	4	1	0	2	4	1	0	2	4	1	1	2
1	0	1	1	1	0	1	1	1	0	1	1	1	0	1	1

Javítsunk a mohó megoldáson: iteratív él-pár csere



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

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

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

Az 1-es városba való visszajutás optimális hossza, ha már meg lett látogatva S és i-ben vagyunk

Dinamikus programozás (1)

Meglátogatott városok halmaza

Utolsó meglátogatott város

	New York	Miami	Dallas	Chicago
1. New York	—	1334	1559	809
2. Miami	1334	—	1343	1397
3. Dallas	1559	1343	—	921
4. Chicago	809	1397	921	—

Triviális esetek: már csak az 1-es város maradt
f(i,{2,3,..,n}) = d[i][1], i=2..n

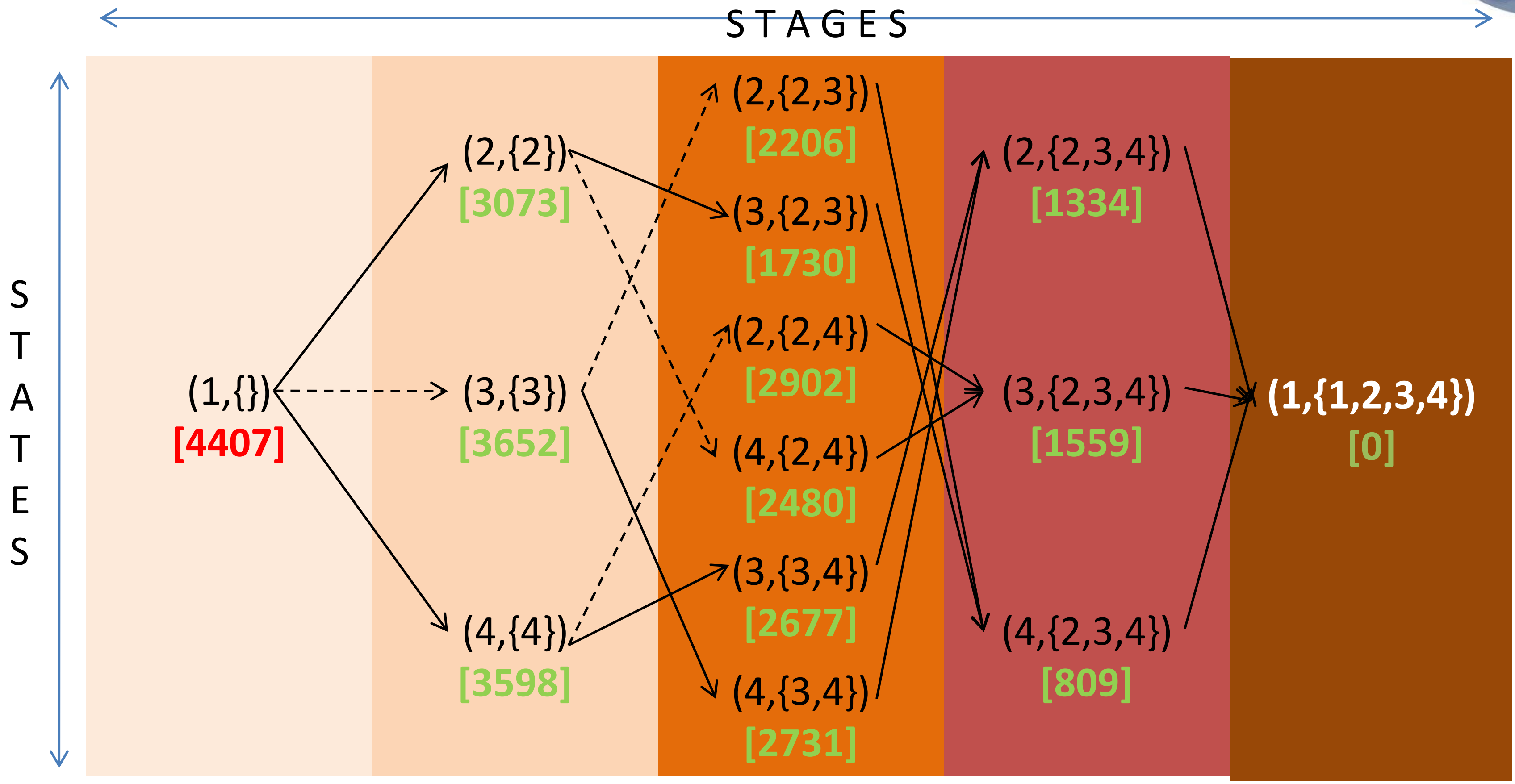
$$f_t(i,S) = \min_{j\neq 1 \text{ and } j\notin S} \{c_{ij} + f_{t+1}(j, S\cup j)\}.$$

	{}	{2}	{3}	{4}	{2,3}	{2,4}	{3,4}	{2,3,4}	{1,2,3,4}
1	4407								0
2		3073			2206	2902		1334	
3			3652		1730		2677	1559	
4				3598		2480	2731	809	

Az indulási, 1-es városban vagyok, és nincs meglátogatott város

Visszaérve az 1-es városba; minden város meglátogatva

Dinamikus programozás (2)



Dimenziók átka (curse of dimensionality)

$$\sum_{k=1}^{n-1}(k*{}_{n-1}C^{k}) + 2 \ll (n-1)!$$

20 város (10. „stage"): állapotok_száma > 1 millió

30 város (15. „stage"): állapotok_száma > 1 billió

100 város (50. „stage"): állapotok_száma > 5.000.000.000.000.000.000.000.000.000.000

```
int TSP(int i, int s, int n, int**d, int**c){
    if(c[i][s] != -1) { return c[i][s]; }
    if (s == ((1<<n) - 2)) { return c[i][s] = d[i][1] + 0; }
    c[i][s] = INF;
    for(j = 1 ; j < n ; ++j){
        if((s & (1<<j)) == 0){
            int temp = TSP(j+1,s|(1<<j),n,d,c);
            if(d[i][j+1] + temp < c[i][s]){
                c[i][s] = d[i][j+1] + temp;
            }
        }
    }
    return c[i][s];
```

c[i][s] már rendelkezésre áll

már csak az 1-es város maradt: $s=2^n-2$

s-nek a j. bitje 0

Csak az 1-estől eltérő városokat tekintem (1..n-1 bitek)

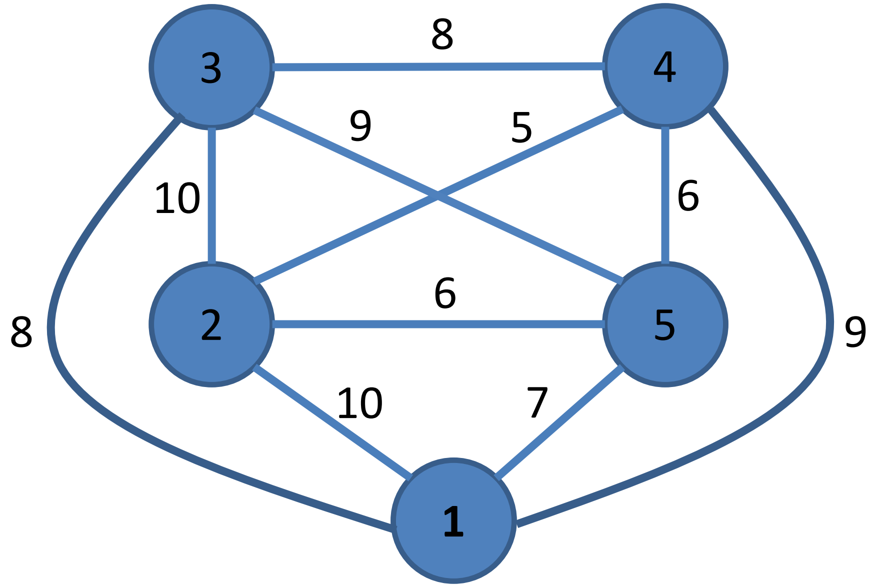
triviális esetek

s-nek a j. bitjét 1-re állítjuk

a j. bitnek megfelelő város a j+1

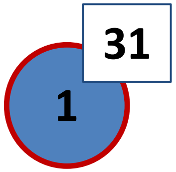
C	{} 0000 0	{2} 0010 2	{3} 0100 4	{4} 1000 8	{2,3} 0110 6	{2,4} 1010 10	{3,4} 1100 12	{2,3,4} 1110 14	{1,2,3,4} 1111 15
1	4407								0
2		3073			2206	2902		1334	
3			3652		1730		2677	1559	
4				3598		2480	2731	809	

Branch and bound (1)



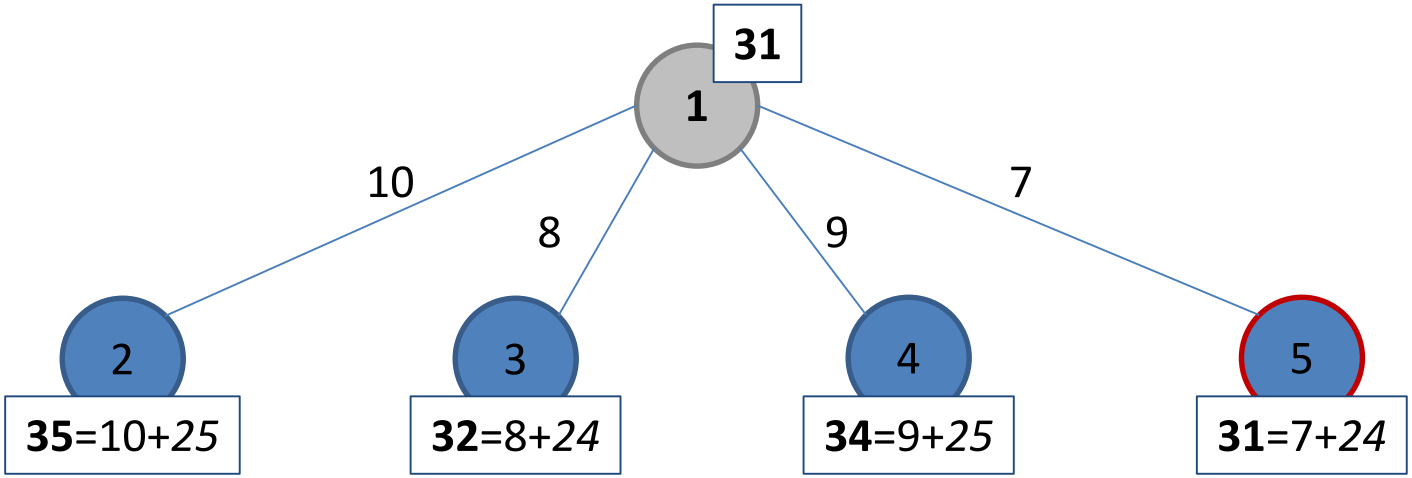
	1	2	3	4	5
1	∞	10	8	9	7
2	10	∞	10	5	6
3	8	10	∞	8	9
4	9	5	8	∞	6
5	7	6	9	6	∞

Branch and bound (2)



	1	2	3	4	5	
1	∞	10	8	9	**7**	**7**
2	10	∞	10	**5**	6	**5**
3	**8**	10	∞	8	9	**8**
4	9	**5**	8	∞	6	**5**
5	7	**6**	9	6	∞	**6**
						31

Branch and bound (3)



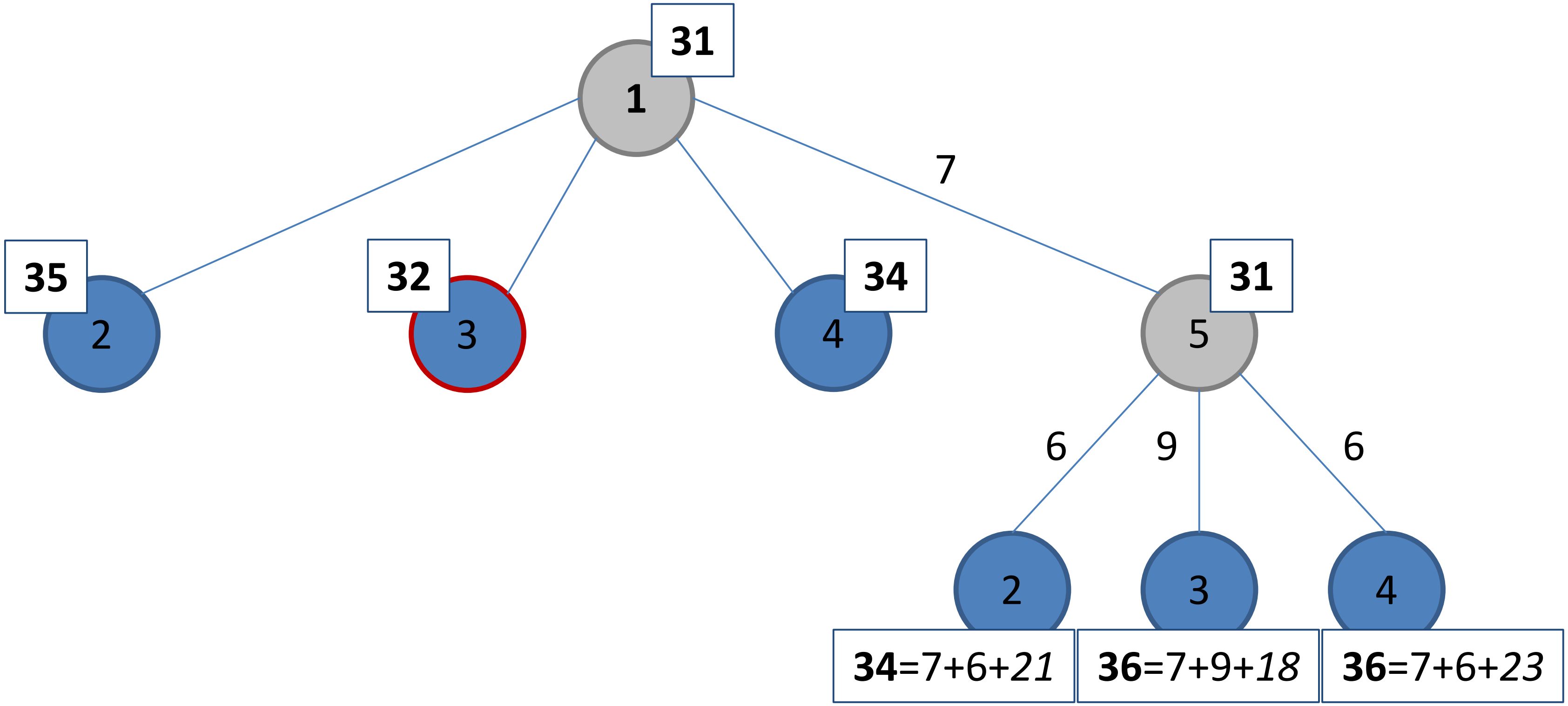
	1	2	3	4	5	
1	∞	∞	∞	∞	∞	∞
2	∞	∞	10	**5**	6	**5**
3	**8**	∞	∞	8	9	**8**
4	9	∞	8	∞	**6**	**6**
5	7	∞	9	**6**	∞	**6**
						25

	1	2	3	4	5	
1	∞	∞	∞	∞	∞	∞
2	10	∞	∞	**5**	6	**5**
3	∞	10	∞	**8**	9	**8**
4	9	**5**	∞	∞	6	**5**
5	7	**6**	∞	6	∞	**6**
						24

	1	2	3	4	5	
1	∞	∞	∞	∞	∞	∞
2	10	∞	10	∞	**6**	**6**
3	**8**	10	∞	∞	9	**8**
4	∞	**5**	8	∞	6	**5**
5	7	**6**	9	∞	∞	**6**
						25

	1	2	3	4	5	
1	∞	∞	∞	∞	∞	∞
2	10	∞	10	**5**	∞	**5**
3	**8**	10	∞	8	∞	**8**
4	9	**5**	8	∞	∞	**5**
5	∞	6	9	**6**	∞	**6**
						24

Branch and bound (4)



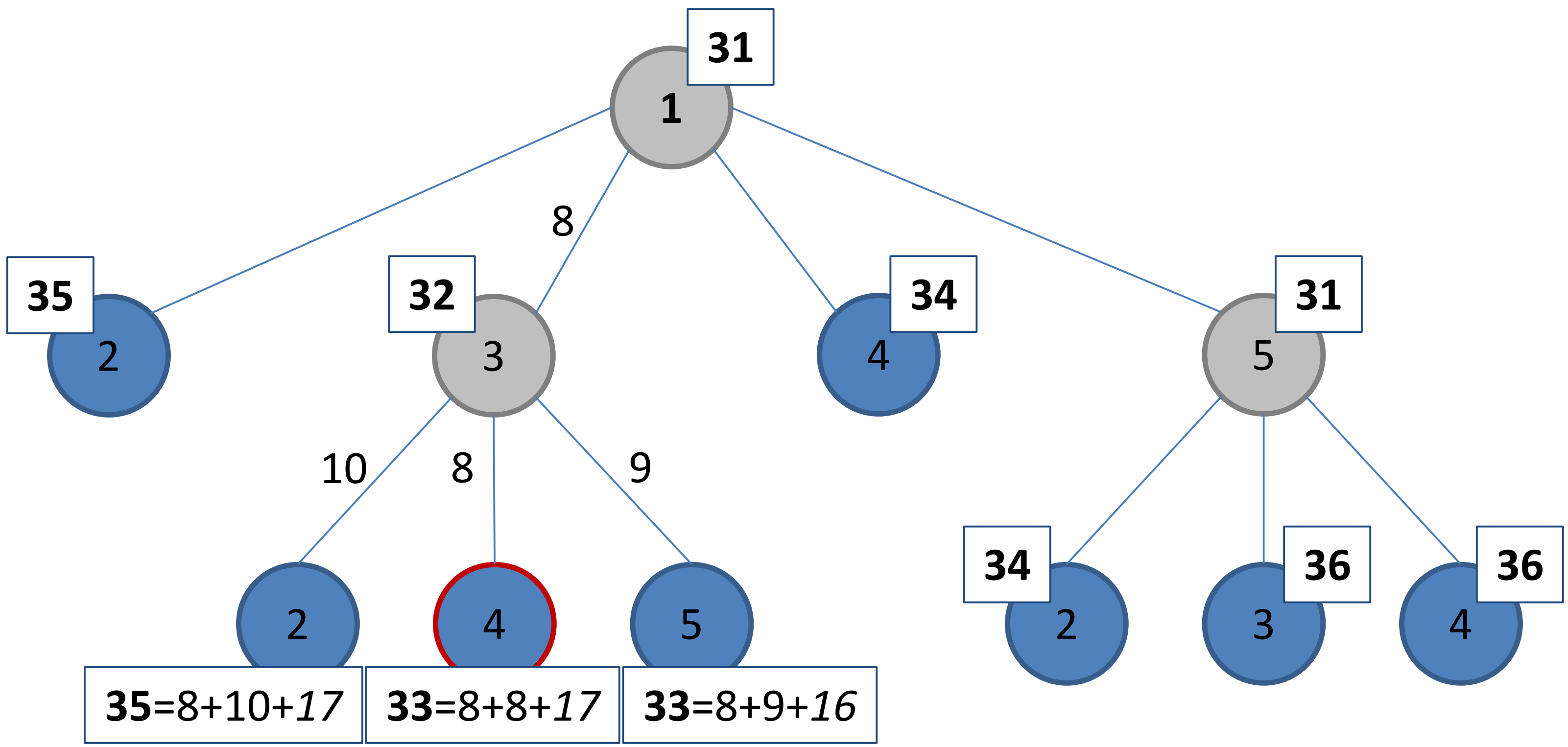
	1	2	3	4	5	
1	∞	∞	∞	∞	∞	∞
2	10	∞	10	**5**	∞	5
3	**8**	10	∞	8	∞	8
4	9	**5**	8	∞	∞	5
5	∞	6	9	**6**	∞	6
					7	*24*

	1	2	3	4	5	
1	∞	∞	∞	∞	∞	∞
2	∞	∞	10	**5**	∞	**5**
3	**8**	∞	∞	8	∞	**8**
4	9	∞	**8**	∞	∞	**8**
5	∞	∞	∞	∞	∞	∞
						21

	1	2	3	4	5	
1	∞	∞	∞	∞	∞	∞
2	10	∞	∞	**5**	∞	**5**
3	∞	10	∞	**8**	∞	**8**
4	9	**5**	∞	∞	∞	**5**
5	∞	∞	∞	∞	∞	∞
						18

	1	2	3	4	5	
1	∞	∞	∞	∞	∞	∞
2	**10**	∞	10	∞	∞	**10**
3	**8**	10	∞	∞	∞	**8**
4	∞	**5**	8	∞	∞	**5**
5	∞	∞	∞	∞	∞	∞
						23

Branch and bound (5)



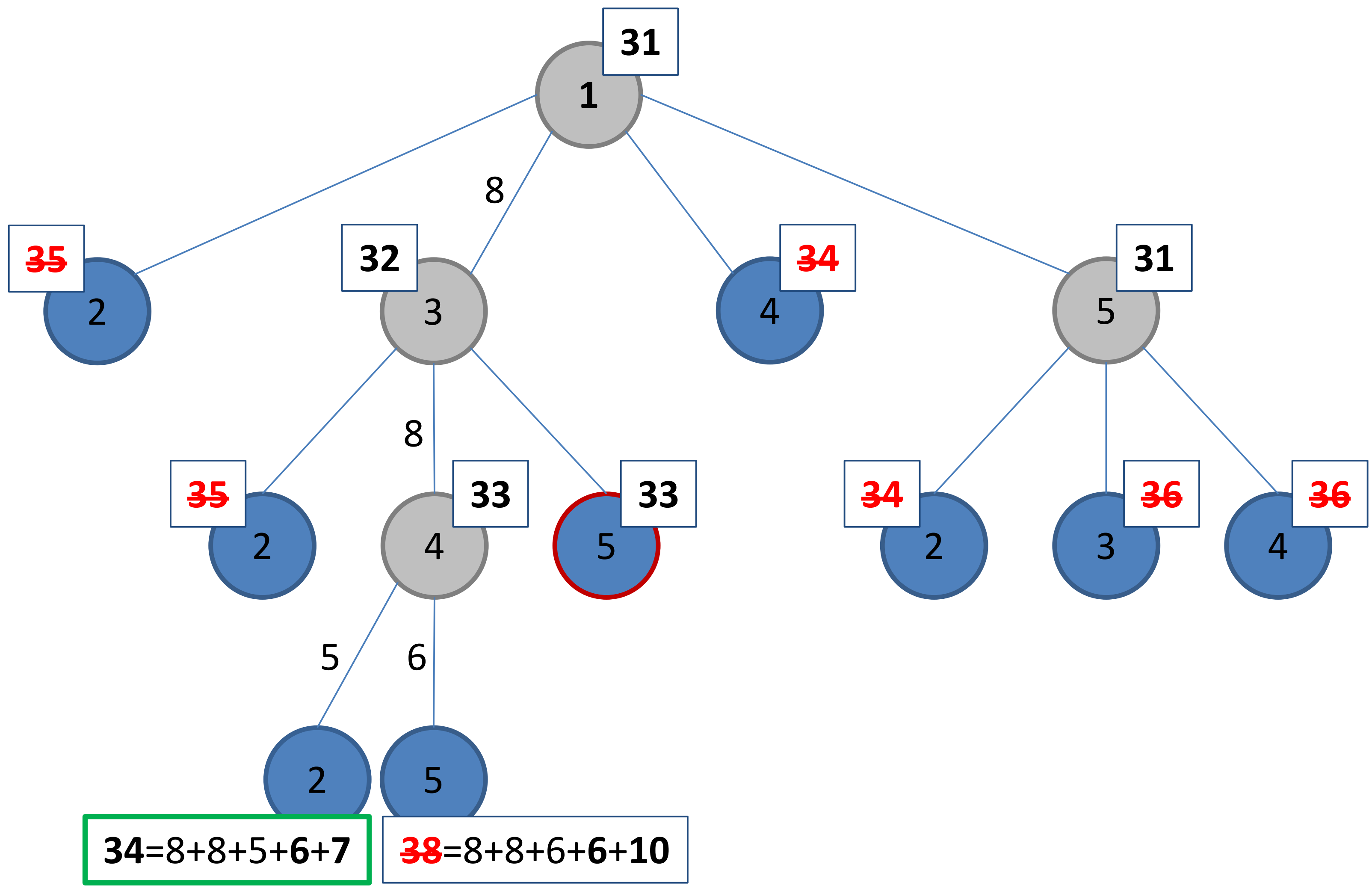
	1	2	3	4	5	
1	∞	∞	∞	∞	∞	∞
2	10	∞	∞	**5**	6	**5**
3	∞	10	∞	**8**	9	**8**
4	9	**5**	∞	∞	6	**5**
5	7	**6**	∞	6	∞	**6**
					8	*24*

	1	2	3	4	5	
1	∞	∞	∞	∞	∞	∞
2	∞	∞	∞	**5**	6	**5**
3	∞	∞	∞	∞	∞	∞
4	9	∞	∞	∞	**6**	**6**
5	7	∞	∞	**6**	∞	**6**
						17

	1	2	3	4	5	
1	∞	∞	∞	∞	∞	∞
2	10	∞	∞	∞	**6**	**6**
3	∞	∞	∞	∞	∞	∞
4	∞	**5**	∞	∞	6	**5**
5	7	**6**	∞	∞	∞	**6**
						17

	1	2	3	4	5	
1	∞	∞	∞	∞	∞	∞
2	10	∞	∞	**5**	∞	**5**
3	∞	∞	∞	∞	∞	∞
4	9	**5**	∞	∞	∞	**5**
5	∞	**6**	∞	6	∞	**6**
						16

Branch and bound (6)

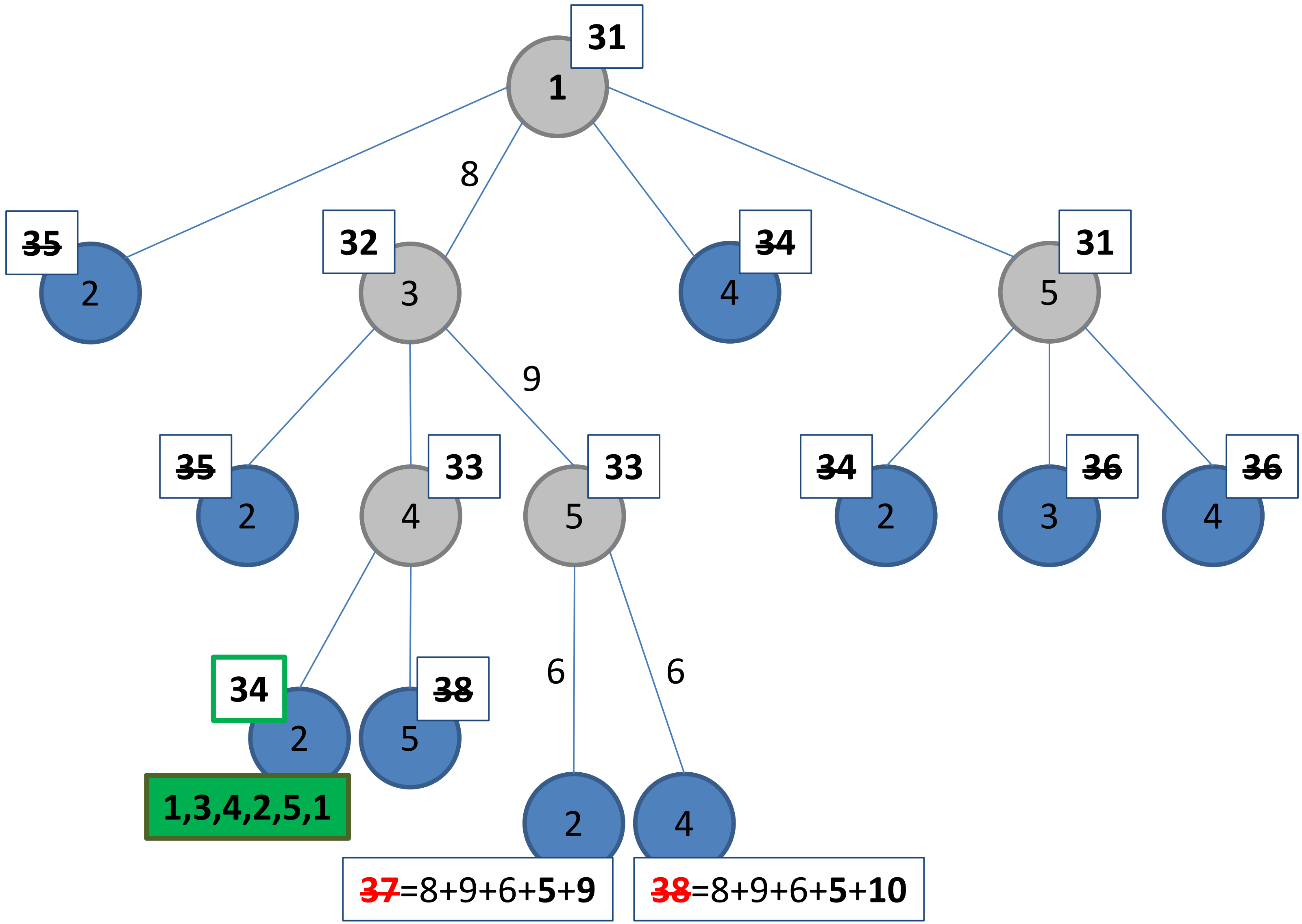


	1	2	3	4	5	
1	∞	∞	∞	∞	∞	∞
2	10	∞	∞	∞	**6**	**6**
3	∞	∞	∞	∞	∞	∞
4	∞	**5**	∞	∞	6	**5**
5	7	**6**	∞	∞	∞	**6**
						17

	1	2	3	4	5	
1	∞	∞	∞	∞	∞	∞
2	∞	∞	∞	∞	**6**	**6**
3	∞	∞	∞	∞	∞	∞
4	∞	∞	∞	∞	∞	∞
5	**7**	∞	∞	∞	∞	**7**
						13

	1	2	3	4	5	
1	∞	∞	∞	∞	∞	∞
2	**10**	∞	∞	∞	∞	**10**
3	∞	∞	∞	∞	∞	∞
4	∞	∞	∞	∞	∞	∞
5	∞	**6**	∞	∞	∞	**6**
						16

Branch and bound (7)



	1	2	3	4	5	
1	∞	∞	∞	∞	∞	∞
2	10	∞	∞	**5**	∞	**5**
3	∞	∞	∞	∞	∞	∞
4	9	**5**	∞	∞	∞	**5**
5	∞	**6**	∞	6	∞	**6**
~~33~~						*16*

	1	2	3	4	5	
1	∞	∞	∞	∞	∞	∞
2	∞	∞	∞	**5**	∞	**5**
3	∞	∞	∞	∞	∞	∞
4	**9**	∞	∞	∞	∞	**9**
5	∞	∞	∞	∞	∞	∞
						14

	1	2	3	4	5	
1	∞	∞	∞	∞	∞	∞
2	**10**	∞	∞	∞	∞	**10**
3	∞	∞	∞	∞	∞	∞
4	∞	**5**	∞	∞	∞	**5**
5	∞	∞	∞	∞	∞	∞
						15

The New York Times

- Mr. Cook and some colleagues have put together a (still unsolved) 1,904,711-point World Traveling Salesman Problem, which takes in every city, town and village in the world, plus a few Antarctic research stations. Meanwhile, the Clay Mathematics Institute is offering a $1 million prize to anyone who can show whether the Traveling Salesman Problem can be fully solved at all, which the mathematician Jordan Ellenberg recently called “the biggest open problem in complexity theory.” (2012)