Bankarev algoritam (Edsger Wybe Dijkstra)

- alokacija resursa i izbjegavanje zaglavljenja

(prva primjena - operacijski sustavi na računalima)

Sigurna i nesigurna stanja

- stanje se smatra sigurnim ako svi procesi koji se trenutno odvijaju mogu završiti svoje izvođenje

Bankarev algoritam određuje da li je stanje sigurno tako da pokušava pronaći redoslijed zahtjeva od strane procesa, koji bi omogućio svakom procesu da dobije sve resurse koji mu nedostaju i da zatim završi izvođenje i vrati sve alocirane resurse sustavu (*alocirani resursi nemogu se otpustiti dok proces nije završio!*).

Ako je moguće pronaći barem jedan takav redoslijed da svi procesi mogu završiti, stanje je sigurno.

Stanje u kojem nije moguće pronaći takav redoslijed zahtjeva je nesigurno stanje.

Primjer vrste problema koji rješava bankarev algoritam

Radionica je dobila zadatak da popravi 4 automobila, A, B, C i D.

Na raspolaganju je 5 autolimara, 2 električara, 4 mehaničara i 3 lakirera.

Na automobilu A, za čiji popravak su potrebna 3 autolimara, 1 električar, 1 mehaničar i 1 lakirer, već rade 2 autolimara, 1 mehaničara i 1 lakirer.

Za popravak automobila B potrebna su 2 električara, 1 mehaničar i 2 lakirera.
Na automobilu već radi 1 elektricar.

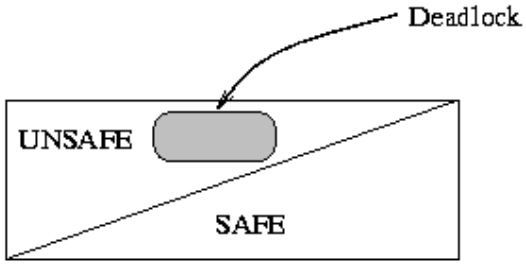
Za popravak automobila C potrebana su 4 autolimara, 2 električara, 1 mehaničar i 1 lakirer. Na automobilu već radi 1 autolimar, 1 mehaničar i 1 lakirer.

Za popravak automobila D potreban je 1 autolimar, 1 električar, 1 mehaničar i 1 lakirer. Na automobilu već radi 1 autolimar, 1 elektricar i 1 lakirer.

Potrebno je pronaći raspored dodjeljivanja radnika kako bi se obavio popravak svih automobila.

Bankarev algoritam predstavlja suboptimalno rješenje => zabranjena su sva nesigurna stanja.

Nesigurno stanje ne vodi nužno u zaglavljenje!



Nedostatak: *svi procesi i njihove potrebe za resursima moraju biti poznati unaprijed*

Formalni zapis algoritma

- **r** - vektor svih resursa u sustavu,
- **p** – vektor svih procesa u sustavu,
- **A –** matrica alociranih resursa po procesima**,**
- **a –** vektor alociranih resursa,
- **N** – matrica potrebnih resursa za završetak procesa,
- **d** – vektor slobodnih resursa.

$$\mathbf{a}(j) = \sum_{i} \mathbf{A}(i, j)$$

a(*j*) – *j*-ta komponenta vektora **a** => suma *j*-tog stupca matrice **A**

$$\mathbf{d} = \mathbf{r} - \mathbf{a}$$

Koraci algoritma

1. Korak

Pronađi redak *i* u matrici **N** koji je manji ili jednak vektoru **d.**

2. Korak

Odredi novu vrijednost vektora **d** kao $\mathbf{d}_{k+1} = \mathbf{d}_k + \mathbf{A}(i)$

3. Korak

Ponavljaj korake 1. i 2. sve dok:
a) svi procese uspješno ne završe svoje izvođenje => početno stanje je sigurno,
b) ne dođe do zaglavljenja (nije moguće obaviti 1. korak) => početno stanje nije sigurno.

Primjer: rješenje problema radionice

r = [5 2 4 3]^T , **p** = [A B C D]^T

$$\mathbf{A} = \begin{bmatrix} 2 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \end{bmatrix} \qquad \mathbf{N} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 2 \\ 3 & 2 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \leftarrow$$

a = [4 2 2 3]^T

N(*D*)=[0 0 1 0] ≤ [1 0 2 0]^T => proces D

Usporedba po komponentama.

d = r – a = [5 2 4 3]^T - [4 2 2 3]^T = [1 0 2 0]^T

$\mathbf{d}_1^T = \mathbf{d}_0^T + \mathbf{A}(D)$ = [1 0 2 0] + [1 1 0 1] = [2 1 2 1] - kraj prve iteracije

$\mathbf{d}_1^T = \mathbf{d}_0^T + \mathbf{A}(D) = [1\ 0\ 2\ 0] + [1\ 1\ 0\ 1] = [2\ 1\ 2\ 1]$ - kraj prve iteracije

2. iteracija

$$\mathbf{N} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 2 \\ 3 & 2 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

$\mathbf{N}(A)=[1\ 1\ 0\ 0] \leq [2\ 1\ 2\ 1]^T$ => proces A

$\mathbf{d}_2^T = \mathbf{d}_1^T + \mathbf{A}(A) = [2\ 1\ 2\ 1] + [2\ 0\ 1\ 1] = [4\ 1\ 3\ 2]$ - kraj druge iteracije

3. iteracija

$$\mathbf{N} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 2 \\ 3 & 2 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

$\mathbf{N}(B)=[0\ 1\ 1\ 2] \leq [4\ 1\ 3\ 2]^T$ => proces B

$\mathbf{d}_3^T = \mathbf{d}_2^T + \mathbf{A}(B) = [4\ 1\ 3\ 2] + [0\ 1\ 0\ 0] = [4\ 2\ 3\ 2]$ - kraj treće iteracije

$\mathbf{d}_3^T = \mathbf{d}_2^T + \mathbf{A}(B) = [4\ 1\ 3\ 2] + [0\ 1\ 0\ 0] = [4\ 2\ 3\ 2]$ - kraj treće iteracije

4. iteracija

$$\mathbf{N} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 2 \\ 3 & 2 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

$\mathbf{N}(C)=[3\ 2\ 0\ 0] \leq [4\ 2\ 3\ 2]^T$ => proces C

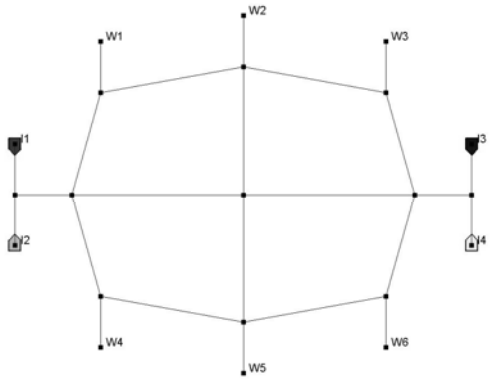
$\mathbf{d}_4^T = \mathbf{d}_3^T + \mathbf{A}(C) = [4\ 2\ 3\ 2] + [1\ 0\ 1\ 1] = [5\ 2\ 4\ 3] = \mathbf{r}$ - kraj proračuna

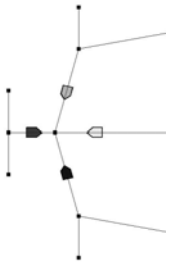
Početno stanje je sigurno!

Vježba: **A**, **r** i **p** ostaju isti; za automobil A potrebna su 3 lakirera => nova matrica **N**

$$\mathbf{N} = \begin{bmatrix} 1 & 1 & 0 & 2 \\ 0 & 1 & 1 & 2 \\ 3 & 2 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Primjer: upravljanje autonomnim viljuškarima u industrijskom postrojenju





Zadatak: potrebno je upravljati autonomnim vozilima koja prenose predmete između radnih stanica (W) tako da ne dođe do zaglavljenja.

Primjer zaglavljenja.

Primjer: upravljanje autonomnim viljuškarima u industrijskom postrojenju

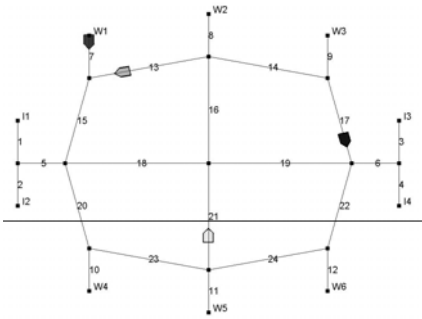
Sustav je modeliran kao graf

- veze između čvorova predstavljaju staze kojima se gibaju vozila
- svaka staza predstavlja jedan *resurs* sustava
- svaka misija vozila (put kojim vozilo prolazi) predstavlja *proces*

Trivijalno rješenje problema – nakon što se vozilu dodijeli misija odredi se put kojim će vozilo prolaziti i ono počinje s gibanjem dok sva ostala vozila miruju => iskoristivost vozila vrlo loša!

Bolje rješenje - bankarevim algoritmom (modificiranim) dodjeljuju se staze (resursi) misijama (procesi) u realnom vremenu =>

dolaskom pred čvor provjerava se sigurnost prelaska na novu granu pozivom bankarevog algoritam



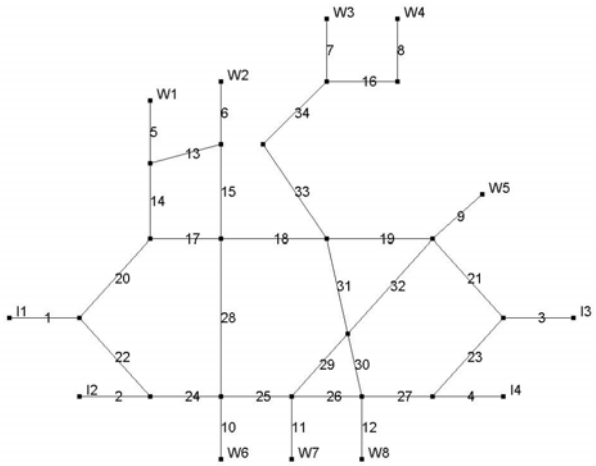
AGV redni br. 1 – crveni
AGV redni br. 2 – zeleni
AGV redni br. 3 – plavi
AGV redni br. 4 - žuti

Vozila se vraćaju u početne stanice (I)

	AGV	Grane											
		1	2	3	4	5	6	7	8	9	10	11	12
	1	**1**	**0**	**0**	**0**	**1**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
	2	**0**	**1**	**0**	**0**	**1**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
	3	**0**	**0**	**1**	**0**	**0**	**1**	**0**	**0**	**0**	**0**	**0**	**0**
N	4	**0**	**0**	**0**	**1**	**0**	**1**	**0**	**0**	**0**	**0**	**0**	**0**
	AGV	13	14	15	16	17	18	19	20	21	22	23	24
	1	**0**	**0**	**1**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
	2	**0**	**0**	**1**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
	3	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
	4	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**0**	**0**	**0**	**0**	**0**

	Grane											
	1	2	3	4	5	6	7	8	9	10	11	12
d	**1**	**1**	**1**	**1**	**1**	**1**	**0**	**1**	**1**	**1**	**1**	**1**
	13	14	15	16	17	18	19	20	21	22	23	24
	0	**1**	**1**	**1**	**0**	**1**	**1**	**1**	**0**	**1**	**1**	**1**

N(*?*) ≤ **d** => AGV ?



standardni BA
- prosječni put svih vozila = 400 m
- prosječno čekanje svih vozila = **47 s**

modificirani BA
- prosječni put svih vozila = 420 m
- prosječno čekanje svih vozila = **18 s**

clips

standardni BA

The shortest path	Vh. 1	Vh. 2	Vh. 3	Vh. 4
Av. trev. dist. [m]	413.25	390.94	398.73	396.31
Av. mis. exec. [s]	36.47	34.88	40.51	41.75
Av. wait. time [s]	15.80	15.33	20.57	21.93
Av. waiting [%]	43.34	43.95	50.78	52.53

modificirani BA

The shortest path	Vh. 1	Vh. 2	Vh. 3	Vh. 4
Av. trev. dist. [m]	413.25	390.94	398.73	396.31
Av. mis. exec. [s]	32.20	32.69	32.61	32.42
Av. wait. time [s]	11.53	13.15	12.68	12.61
Av. waiting [%]	35.82	40.21	38.87	38.89

Splitting a path	Vh. 1	Vh. 2	Vh. 3	Vh. 4
Av. trev. dist. [m]	415.23	424.40	433.44	407.04
Av. mis. exec. [s]	25.25	26.43	26.09	24.97
Av. wait. time [s]	4.49	5.21	4.42	4.61
Av. waiting [%]	17.77	19.72	16.92	18.48