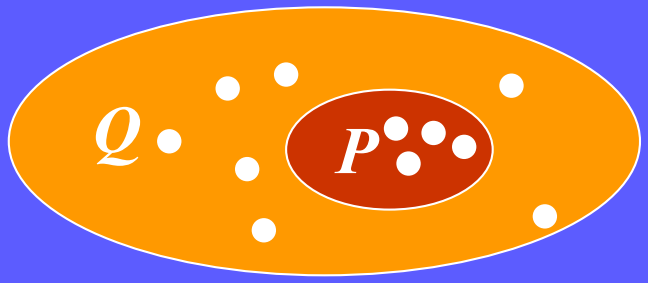
Algorithms

Reductions

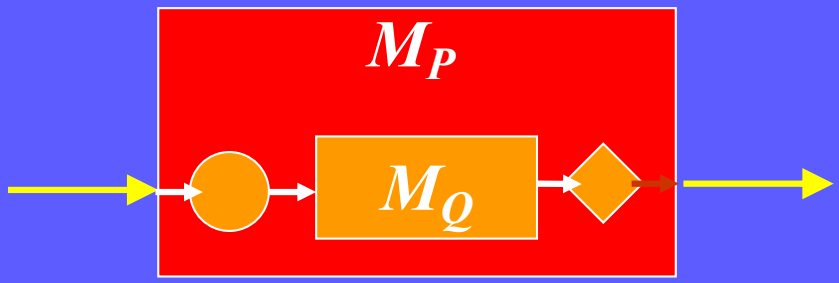
(pp. 321~326)

Reductions

- **Reductions**
  - **Problem *P* is a special case of an solved problem *Q***
  - **A solution of problem *P* using a black box that soles problem *Q***
  - **Goals**
    - **A solution of *P* that uses a black box for *Q* can be translated into an algorithm for M_P if we know an algorithm M_Q for Q**



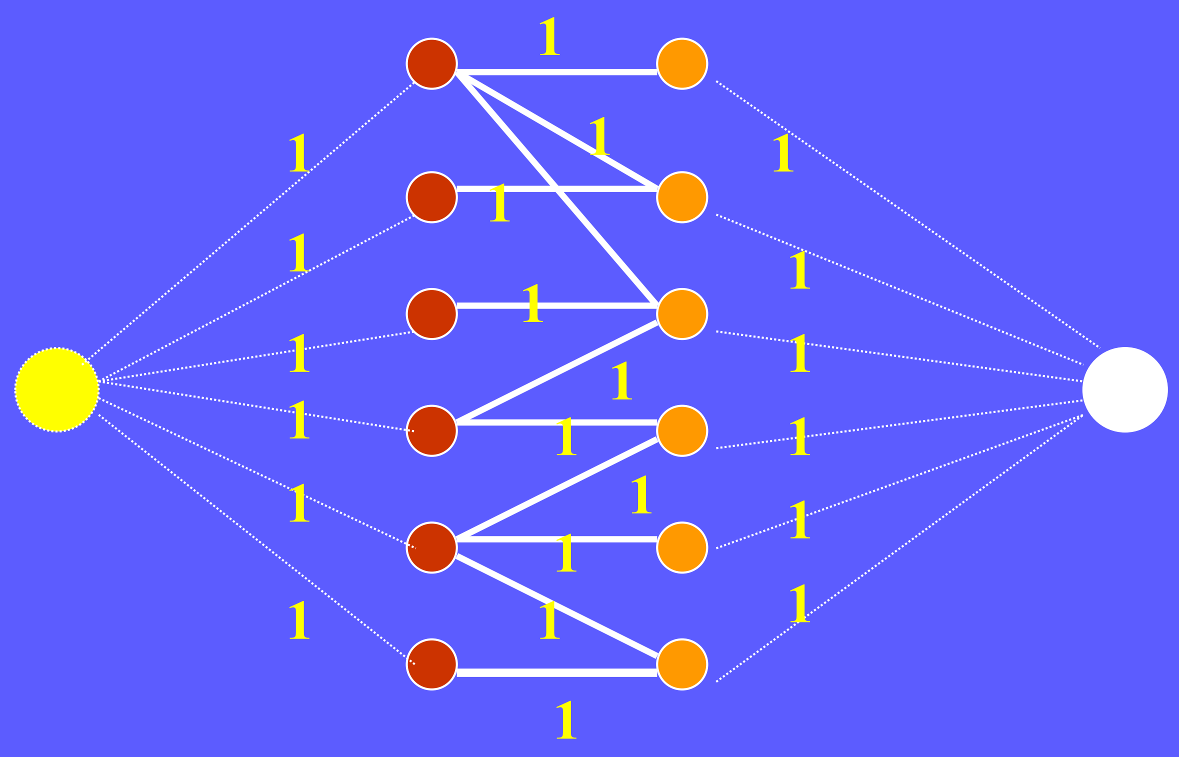
Problem Instance



Algorithm

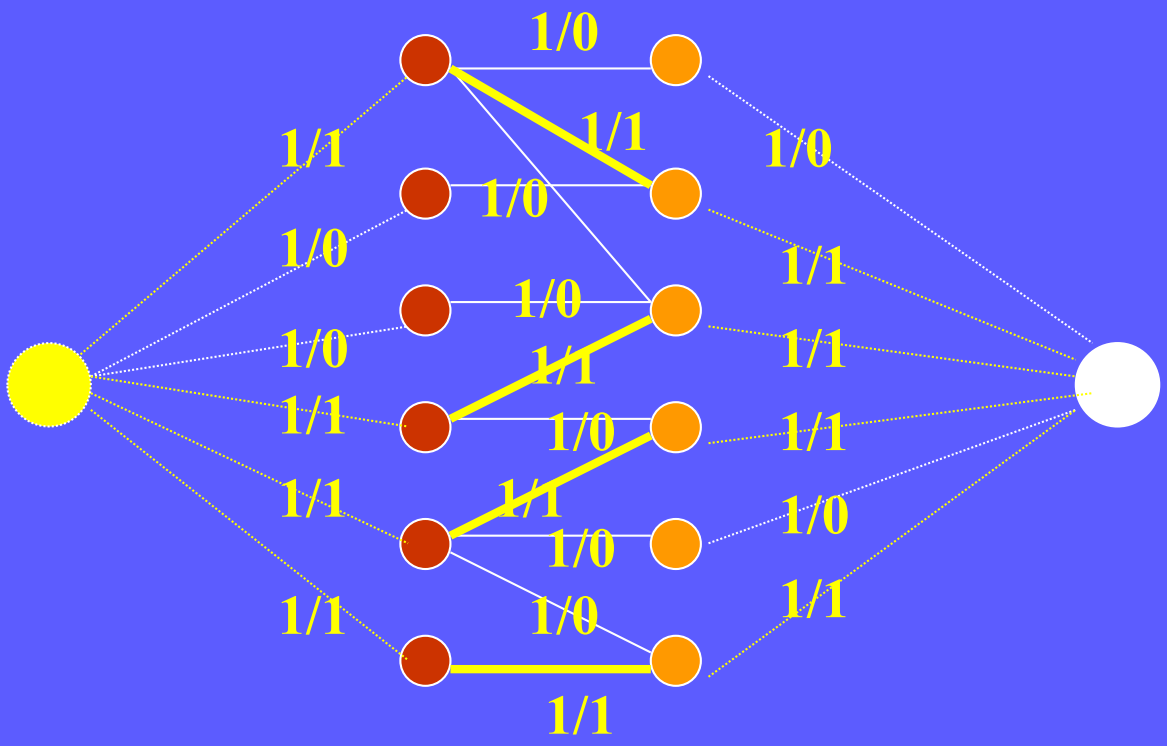
Observation: Reducing Partite Graph to Network Flow

- **M is a maximum matching iff the corresponding flow is a maximum flow in G'**



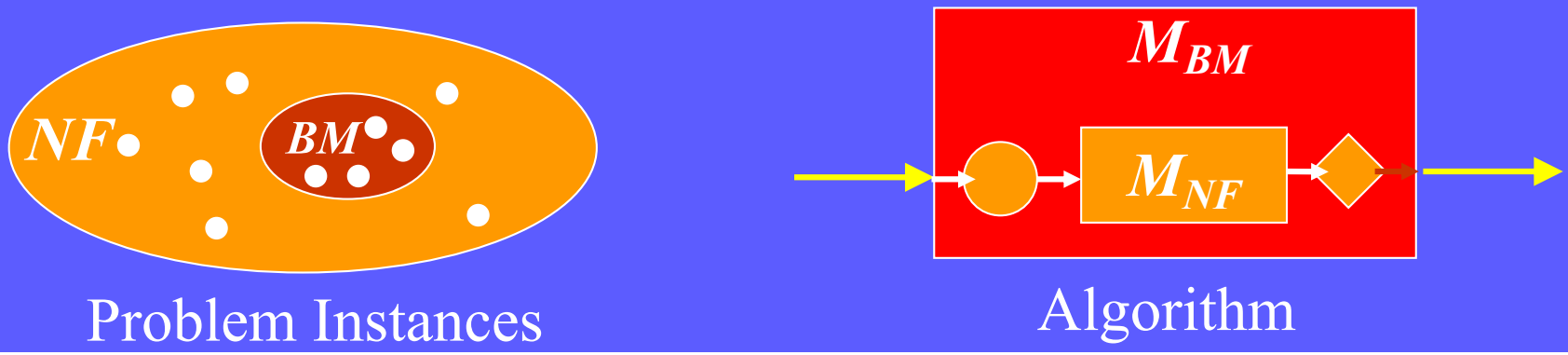
Reducing Partite Graph to Network Flow

- **M is a maximum matching**
  iff the corresponding flow is a maximum flow in G'



Reductions

- **Reductions**
  - **Bipartite Matching is a special case of Network Flow**
  - **A solution of problem *Bipartite Matching* using a black box that solves Network Flow**
  - **Goals**
    - **A solution of *Bipartite Matching* that uses a black box for *Network Flow* can be translated into an algorithm for M_{BM} if we know an algorithm M_{NF} for Network Flow**



Cyclic String Matching

Reductions

Cyclic String Matching

- Let $A=a_0a_1\ldots a_{n-1}$ and $B=b_0b_1\ldots b_{n-1}$
  be two strings of size n,
  Determine whether B is a cyclic shift of A

  e.g. A = abcdefghijk, B = defghijkabc
  B is a cyclic shift of A

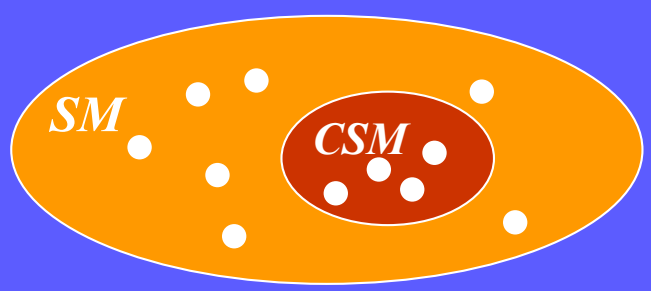
Cyclic String Matching (cont.)

- **Reduction**
  - **Cyclic String Matching as a special case of String Matching**
  - **Let Text=AA, Pattern=B**

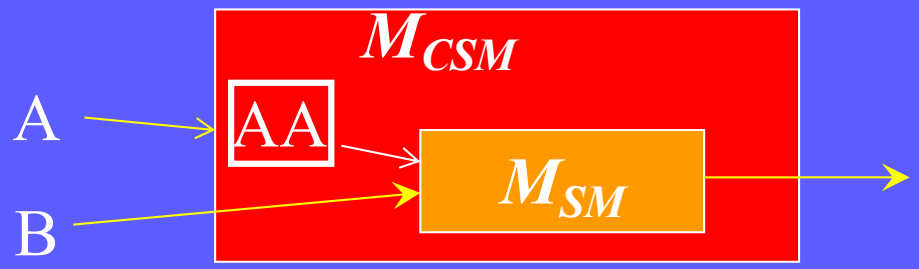
    B is a cyclic shift of A if B is a substring of AA

    e.g. B = defghijkabc is a cyclic shift of A = abcdefghijk

    B is a substring of AA= abcdefghijkabcdefghijk
  - **M_{SM} solve in linear time => linear time algorithm for M_{CSM}**



Problem Instances



Algorithm

Systems of Distinct Representatives

Reductions

Systems of Distinct Representatives

- Let $S_1S_2\ldots S_k$ be a collections of sets,
  a system of distinct representatives (SDR)
  is a set $R=\{r_1r_2\ldots r_k\}$
  such that, $r_i \in S_i$, $\forall$ i, $1\leq i \leq k$
  (R includes exactly one representative from each set)
- Example 1
  S1={1, 2}, S2={2, 3, 4}, S3={1, 3}, S4={1, 2, 3}
  SDR={1, 4, 3, 2}
- Example 2
  S1={1, 2}, S2={2, 3, 4}, S3={1, 3}, S4={1, 2, 3}, S5={2, 3}
  SDR?

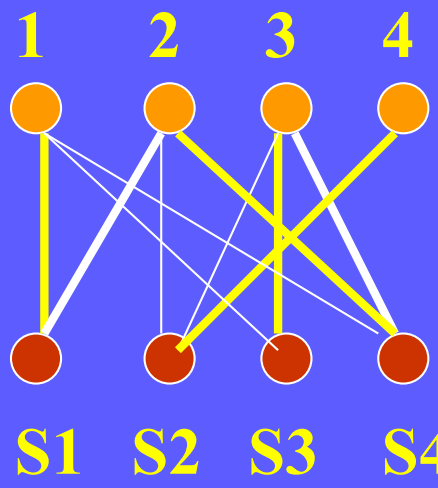
Problem of SDR

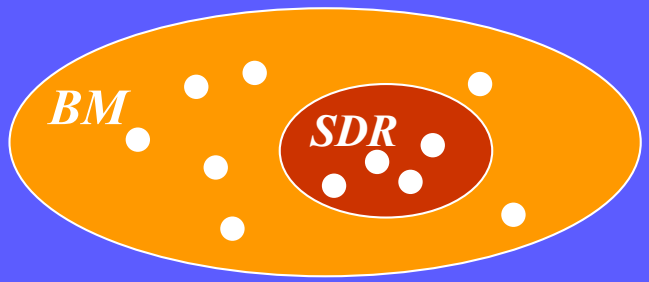
- **Given a finite collection of finite sets,**
  find an SDR for the collection, or
  determine that none exists.
- **Hall's Theorem**
  Let $S_1, S_2, ..S_k$ be a collection of sets.
  This collection has an SDR
  iff for every subset $\{i_1, i_2, \ldots, i_m\}$ of $\{1, 2, 3, \ldots, k\}$
  cardinality$[S_{i1} \cup S_{i2} \cup ... \cup S_{im}] \geq m$
- **Example 2:**
  $S_1=\{1, 2\}$, $S_2=\{2, 3, 4\}$, $S_3=\{1, 3\}$, $S_4=\{1, 2,3\}$, $S_5=\{2, 3\}$
  card$[S_1 \cup S_3 \cup S_4 \cup S_5] = 3 < 4$

Reduction of SDR

- **SDR as a special case of bipartite matching problem**
- **Let G=(V, U, E)**
  - **vertex v_i in V for each set S_3**
  - **vertex u_j in U for each possible element**
  - **edge (v_i, u_j) E iff $u_j \in S_i$**

  an SDR is a matching in G of size k.
- **Example: S1={1, 2}, S2={2, 3, 4}, S3={1, 3}, S4={1, 2, 3}**





Problem Instances

Algorithm

Sequence Comparison

Reductions

Sequence Comparisons

- **Editing distance between two strings**
  - **insertions**
  - **deletions**
  - **replacements**

		b	a	b	b
	(0)	(1)	2	3	4
a	1	1	(1)	2	3
b	2	1	2	(1)	2
b	3	2	2	2	(1)
c	4	3	3	3	(2)

Sequence Comparisons

- Reduce to single source shortest path problem
- Each entry corresponds to a vertex in the graph
- Edges
  - horizontal edges correspond to insertions
  - vertical edges correspond to deletions
  - diagonal edges correspond to replacements
- Cost of edge is 1

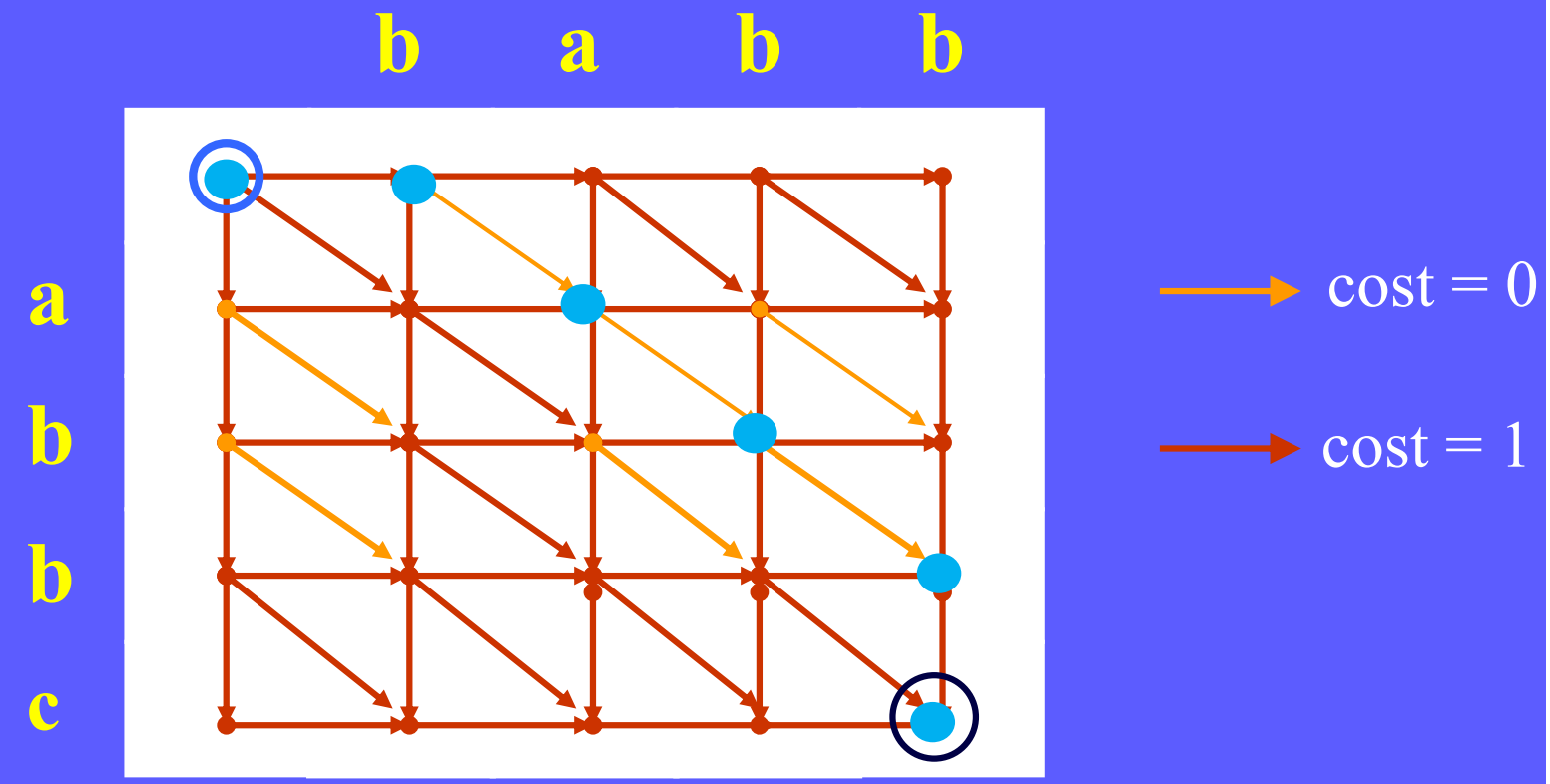
  except for diagonal edges

  that correspond to equal characters

  (no replacement is necessary) whose cost is 0.

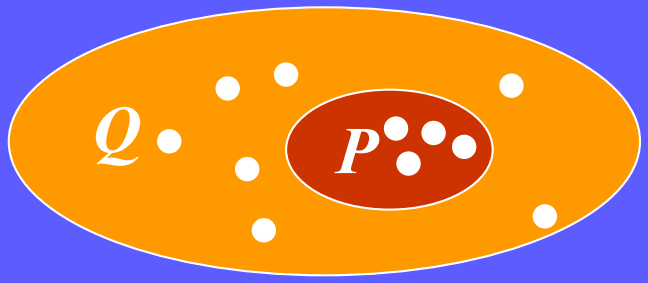
Sequence Comparisons

- Editing distance between abbc & babb
- abbc -> babbc -> babb

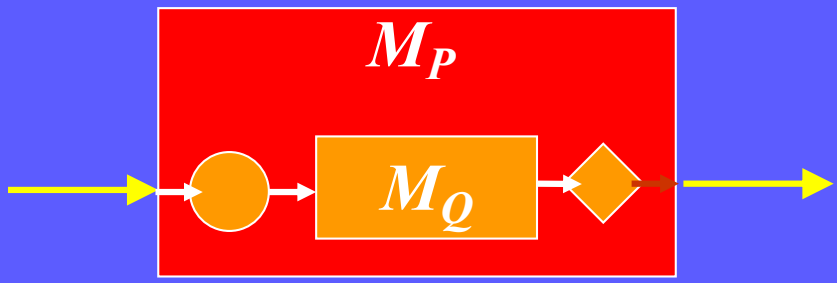


Conclusions

- **Reductions**
  - **Problem *P* is a special case of an solved problem *Q***
  - **A solution of problem *P* using a black box that soles problem *Q***
  - **Goals**
    - **A solution of *P* that uses a black box for *Q* can be translated into an algorithm for M_P if we know an algorithm M_Q for Q**
    - **If *P* is known to be a hard problem (or if we know the lower bound for *P*), then the same lower bound may be applied to *Q***
    - **Solving *P* cannot be harder than solving *Q***



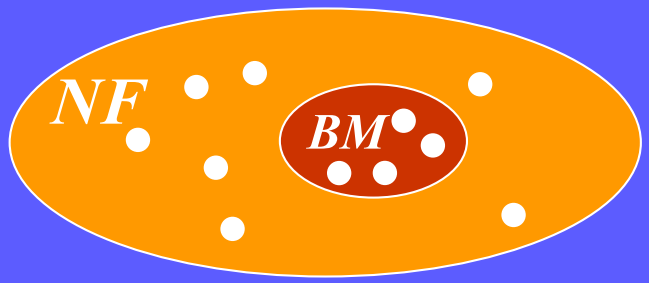
Problem Instance



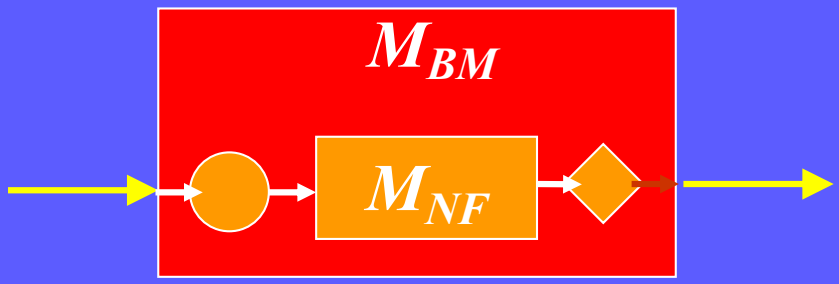
Algorithm

Conclusions (cont.)

- **Reductions**
  - **Bipartite Matching is a special case of Network Flow**
  - **A solution of problem *Bipartite Matching* using a black box that solves Network Flow**
  - **Goals**
    - **If *Bipartite Matching* is known to be a hard problem (or if we know its lower bound), then the same lower bound may be applied to *Network Flow***
    - **Solving *Bipartite Matching* cannot be harder than solving *Network Flow***



Problem Instances



Algorithm