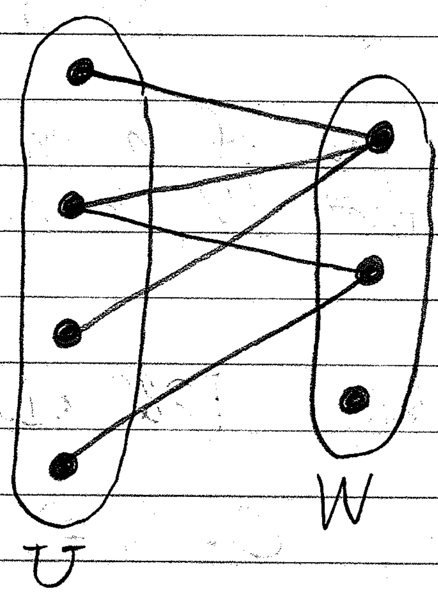
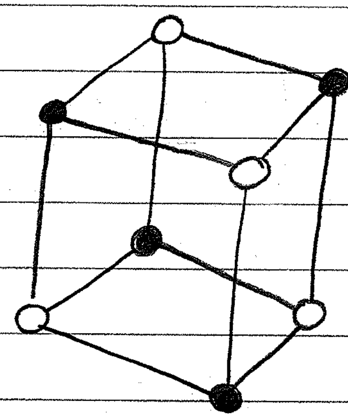
22 June

Bipartite matching [P&S §10.1 — 10.3]

<u>Defn.</u> A graph $G=(V,E)$ is <u>bipartite</u> if there exists a partition of the vertex set V into two sets U and W such that every edge $e \in E$ has one endpoint in U and one endpoint in W.

<u>Examples</u>





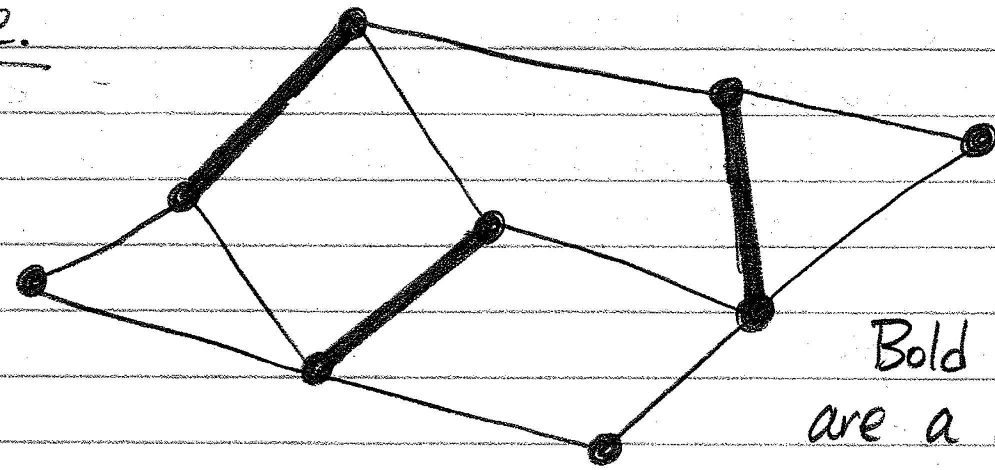
$U = \{\bullet\}$ $W = \{\circ\}$

<u>Proposition</u> [P&S Prop. 1, §A.2, page 21]
A graph is bipartite if and only if it has no cycle of odd length.

<u>Proof.</u> Exercise.

Defn. A matching M of a graph $G=(V,E)$ is a subset of the edge set E such that no two edges of M are incident upon the same vertex.

Example.



Bold edges are a matching.

Defn. A maximum matching of a graph G is a matching M such that $|M|$ is maximized over all matchings of G.

Maximum matching problem: [P&S §10.1]

Given: An undirected graph $G=(V,E)$.

Goal: Find a maximum matching of G.

Applications: Matching candidates to jobs, pairing compatible people into teams, matching med students to hospital residencies, etc.

Matching — 2

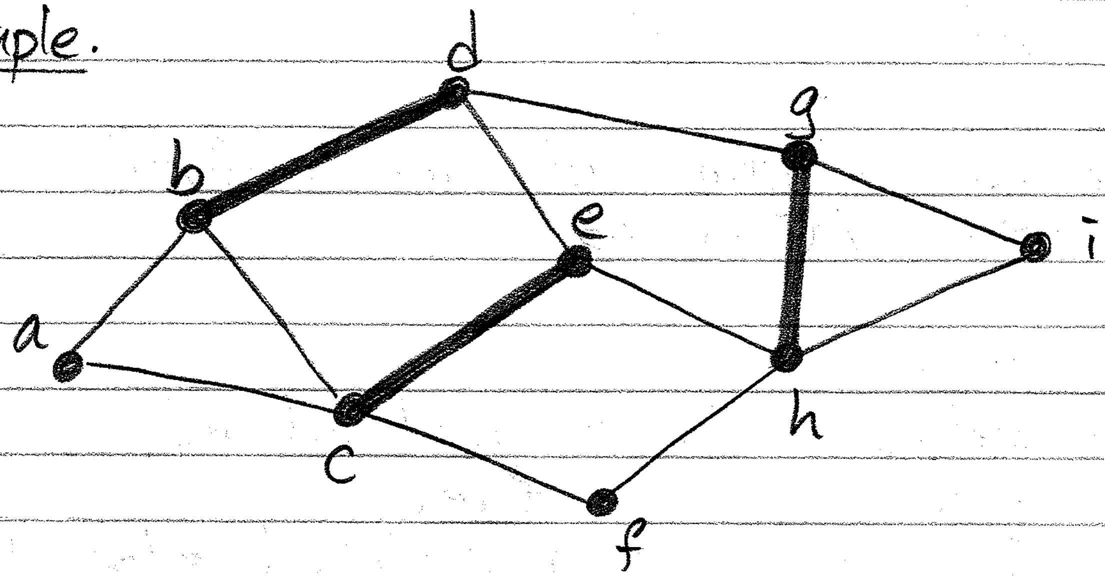
22 June

Consider a graph G and a fixed matching M of G.

Terminology:

- Edges in M are <u>matched</u> edges; other edges are <u>free</u>.
- If $\{u,v\}$ is a matched edge, then v is the <u>mate</u> of u, and vice versa.
- Vertices that are endpoints of matched edges are <u>matched</u> vertices; other vertices are <u>exposed</u>.
- A path $P=(v_1, v_2, \dots, v_k)$ is <u>alternating</u> if $\{v_1,v_2\}, \{v_3,v_4\}, \dots, \{v_{2j-1}, v_{2j}\}, \dots$ are free, while $\{v_2,v_3\}, \{v_4,v_5\}, \dots, \{v_{2j}, v_{2j+1}\}, \dots$ are matched. (So the edges go free, matched, free, matched, ... along the path.)
- For a vertex v, if there exists an alternating path P (with respect to M) such that the first vertex in P is exposed and v has odd index in P (i.e., v is the first, third, fifth, ... vertex in P), then v is <u>outer</u> (with respect to M). If no such path exists, v is <u>inner</u>.
- An alternating path $P=(v_1, v_2, \dots, v_k)$ is <u>augmenting</u> if both v_1 and v_k are exposed.

Example.



(a, b, d, e, c, f) is an augmenting path. So are (f, h, g, i) and (i, g, h, e, c, a).

Observe: Every augmenting path must be of odd length (that is, an even number of vertices and an odd number of edges).

Matching — 3

22 June

[P&S Lemma 10.1]

<u>Lemma.</u> Let P be the set of edges on an augmenting path $(v_1, v_2, \ldots, v_{2k})$ in a graph G with respect to the matching M. Then $M' = M \oplus P$ is a matching of cardinality $|M'| = |M| + 1$.

<u>Note:</u> $M \oplus P \overset{\text{def}}{=} (M \setminus P) \cup (P \setminus M)$.
<u>Symmetric difference.</u> (Also written $M \Delta P$.)

<u>Proof.</u> First, $M \oplus P$ is a matching:
Suppose for the sake of contradiction that two edges $e, e' \in M \oplus P$ are incident upon the same vertex v.

<u>Case 1:</u> $e, e' \in M \setminus P$. But then M is not a matching.

<u>Case 2:</u> $e, e' \in P \setminus M$. Because P is alternating, edges in $P \setminus M$ are of the form $\{v_{2j-1}, v_{2j}\}$, so two of them cannot be incident upon the same vertex.

<u>Case 3:</u> WLOG, $e \in M \setminus P$ and $e' \in P \setminus M$. Since $e \in M$ and v is an endpoint of e, v is not exposed, so it is not v_1 or v_{2k}. Say $v = v_i$. Then one of the edges $\{v_{i-1}, v_i\}$ or $\{v_i, v_{i+1}\}$ is in M, because P is alternating. WLOG, $\{v_i, v_{i+1}\} \in M$.

Now $e \in M \setminus P$, so $e \neq \{v_i, v_{i+1}\}$ because e is not in P. But this means that M contains two distinct edges, e and $\{v_i, v_{i+1}\}$, incident upon the same vertex v, so M is not a matching.

We have reached a contradiction in all three cases. Therefore $M \oplus P$ cannot have two edges incident upon the same vertex, so it is a matching.

Now, P contains $2k-1$ edges: k of them are free (the edges $\{v_1, v_2\}, \{v_3, v_4\}, \ldots, \{v_{2k-1}, v_{2k}\}$) and the remaining $k-1$ belong to M. So $M \oplus P$, which "flips" the status of all edges of P from free to matched or vice versa, has exactly one more edge than M. $\square$

Matching — 4

22 June

Theorem [P&S Thm 10.1] A matching M in a graph G is maximum if and only if there is no augmenting path in G with respect to M.

Proof. If there is an augmenting path P with respect to M, then $M \oplus P$ is a strictly larger matching by the previous lemma, so M is not maximum.

Conversely, if M is not maximum, then there exists a matching M' such that $|M'| > |M|$. Consider the subgraph $H = (V, M \oplus M')$. Because M and M' are matchings, and every edge in H is in M or M', no vertex can have degree more than 2 in H, and if a vertex has degree 2 then one of the incident edges is in M and the other is in M'. So all connected components of H are paths or cycles of even length. Such a cycle has equally many edges from M as from M'. But $|M'| > |M|$, so some connected component of H must contain more edges from M' than from M, and this connected component must be a path — in particular, it is an augmenting path with respect to M.

$\square$

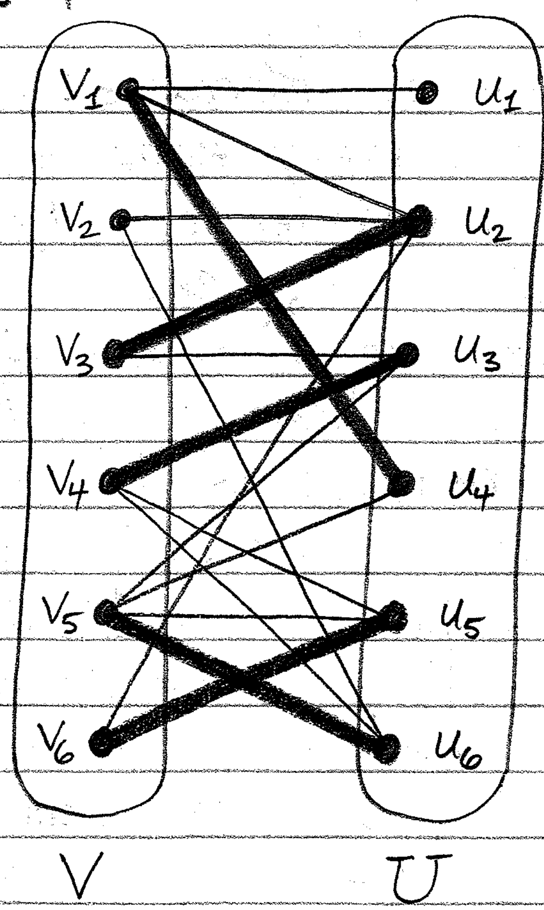
Bipartite matching via augmenting paths

Idea:

1. Start with an empty matching.
2. Find an augmenting path. If none exists, current matching is optimal.
3. "Flip" all edges along augmenting path from free to matched or vice versa.
4. Go back to step 2.

Q: Given a matching M in a bipartite graph G, how do we search for an augmenting path? (For step 2 above.)

Example.



Bipartite matching via augmenting paths — 2

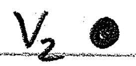
22 June

Observations.

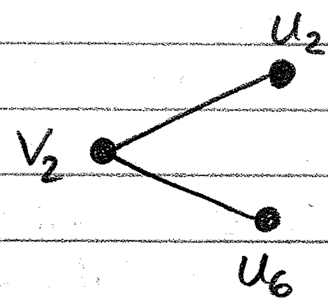
- Any augmenting path must start at an exposed vertex in V and end at an exposed vertex in U (or vice versa, but the reverse of an augmenting path is also an augmenting path). So we can search for augmenting paths by exploring outward from exposed vertices in V, following alternating paths.
- If we follow an alternating path starting at a vertex in V, then we will always take free edges rightward (from V to U) and matched edges leftward (from U to V).
- There is never any choice in the leftward steps, because each vertex in U is an endpoint of at most one matched edge — in our alternating paths, each matched vertex in U will be immediately followed by its mate, which is a vertex in V.
- Our goal is to follow one of these alternating paths to an exposed vertex in U, because then we have an augmenting path.

In the example, the only exposed vertex in V is v_2. So we start there and explore outward, following alternating paths, seeking an exposed vertex in U. We can do this via <u>breadth-first search</u>:

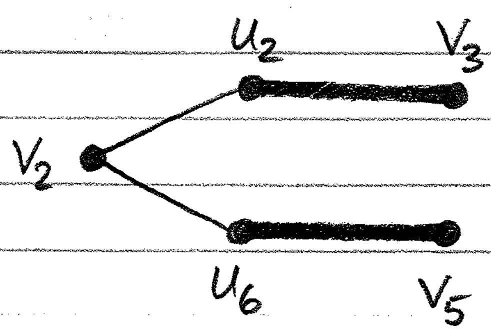
1. Start at v_2. (If there are several exposed vertices in V, we can start at all of them simultaneously.)



2. See where we can go from v_2 by following <u>free</u> edges. We can go to u_2 or u_6:

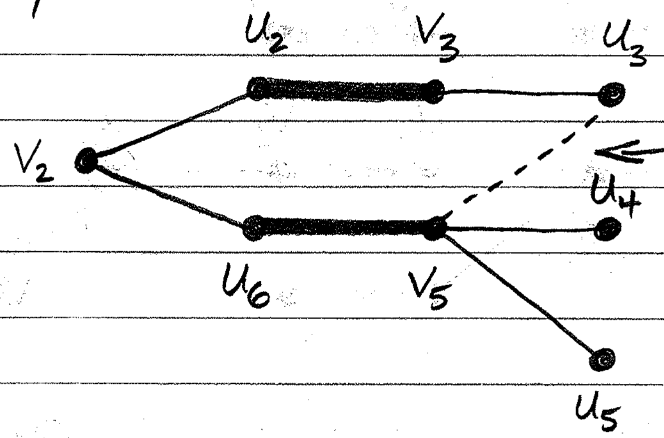


3. See where we can go from u_2 and u_6 by following <u>matched</u> edges. We can go to v_3 or v_5:



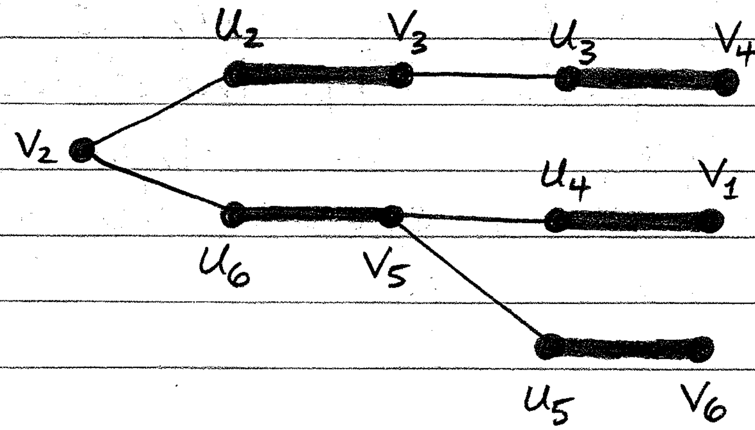
22 June

4. See where we can go from v_3 and v_5 by following <u>free</u> edges. We can go to u_3 from v_3, and to u_3, u_4, and u_5 from v_5. But we don't gain anything from having more than one way to get to u_3, so we need only record one way in the search tree we are building:

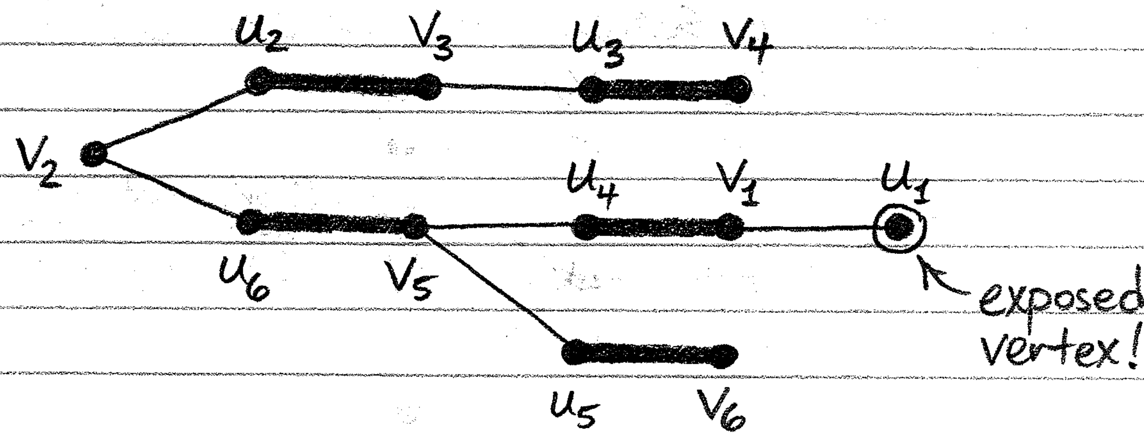


(Don't need multiple ways to get to u_3. One way is enough.)

5. See where we can go from u_3, u_4, and u_5 by following <u>matched</u> edges. From u_3 we can go to v_4, from u_4 we can go to v_1, and from u_5 we can go to v_6:



6. See where we can go from v_4, v_1, and v_6 by following <u>free</u> edges. From v_4 we can go to u_5 or u_6 (but we've visited both of them already), from v_1 we can go to u_1 or u_2 (but we've been to u_2 already), and from v_6 we can go to u_2 (been there). So the only <u>new</u> vertex we can reach is u_1, from v_1:



7. See where we can go from u_1 by following <u>matched</u> edges. But we can't go anywhere, because u_1 is an exposed vertex. This means that we have reached our goal! The path $v_2 - u_6 - v_5 - u_4 - v_1 - u_1$ is an augmenting path. Flip the status of all edges along this path from free to matched or vice versa to get a larger matching.

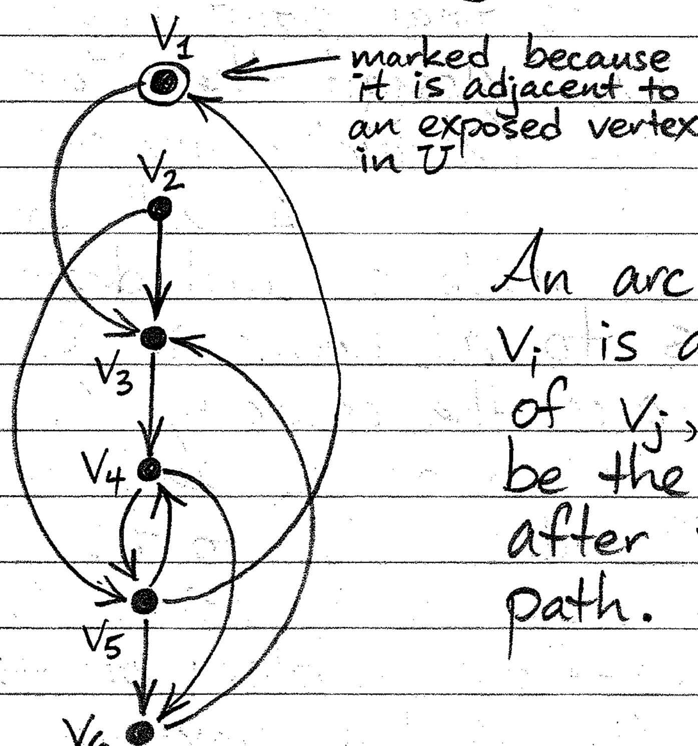
Bipartite matching via augmenting paths — 4

22 June

A little efficiency improvement.

Because any matched node in U will be immediately followed by its mate, we don't have to include the nodes in U in this search — we can just go from a node in V directly to another node in V. To facilitate this, we construct the auxiliary digraph:

(This is the front cover of P&S.)



An arc (v_i, v_j) means that v_i is adjacent to the mate of v_j, so that v_j can be the next vertex in V after v_i in an alternating path.

Then the goal of our search becomes finding a directed path in the auxiliary digraph from an exposed node in V to a node that is adjacent to an exposed node in U; in this case, $v_2 \rightarrow v_5 \rightarrow v_1$. See P&S for more implementation details.

Bipartite matching via max flow [P&S §10.3]

We can *reduce* the bipartite matching problem to the max-flow problem, which is to say that we show how to solve the bipartite matching problem efficiently by using an algorithm that solves the max-flow problem efficiently.

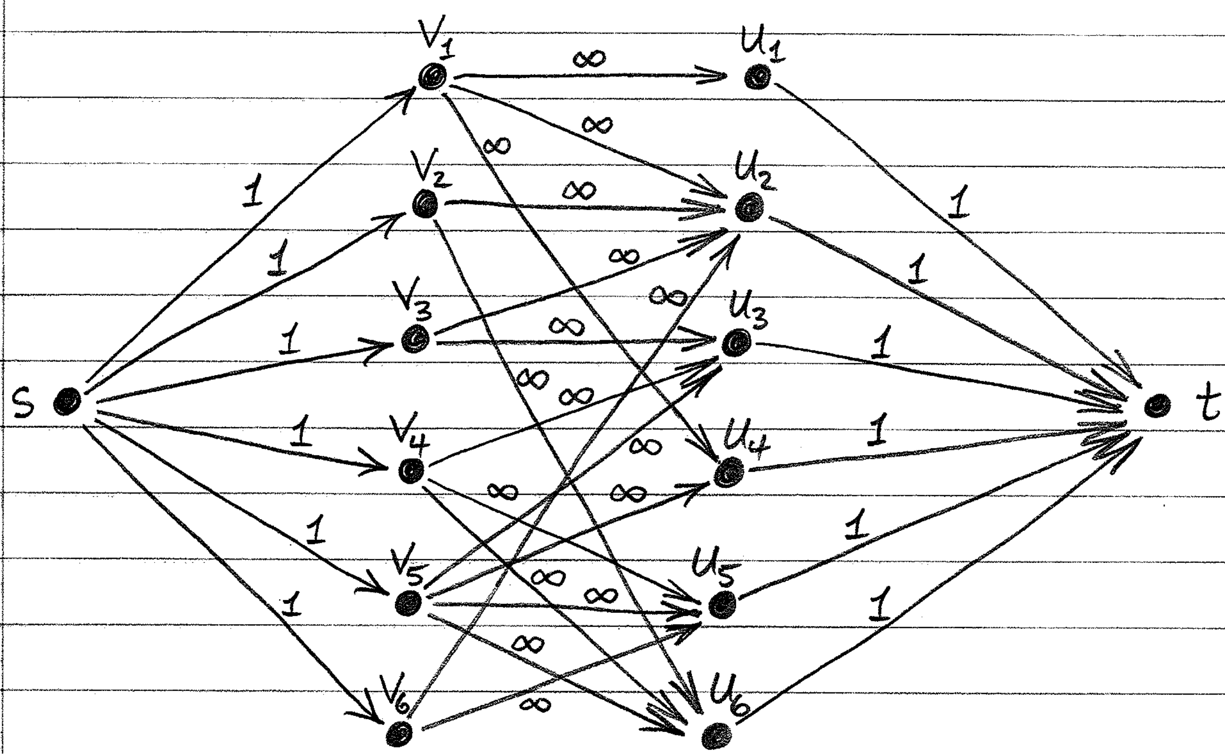
Idea:

1. Convert all edges in the bipartite graph into directed arcs that point rightward from V to U. Assign these arcs the capacity ∞. (Actually any capacity greater than or equal to 1 will work.)
2. Add a source node s, and draw arcs from s to all nodes in V. Assign these arcs the capacity 1.
3. Add a terminal node t, and draw arcs from all nodes in U to t. Assign these arcs the capacity 1.
4. Solve the max-flow problem on this flow network (i.e., find a maximum $s-t$ flow) using, say, Ford-Fulkerson (so that all flows along arcs will be integers).
5. The $V \to U$ arcs will have flows that are either 0 or 1 (why?), and the arcs with flow 1 will correspond to a maximum matching in the original bipartite graph (why?).

22 June

Bipartite matching via max flow — 2

For the previous example, the flow network looks like this:



Note: The maximum s–t flow here can also be found by using the simplex algorithm to solve the max-flow LP; all arc flows will be integers. Choosing ∞ for the capacities of the $V \to U$ arcs means that this LP doesn't need capacity constraints for those arcs.