Eidgenössische Technische Hochschule Zürich
Ecole polytechnique fédérale de Zurich
Politecnico federale di Zurigo
Swiss Federal Institute of Technology Zurich

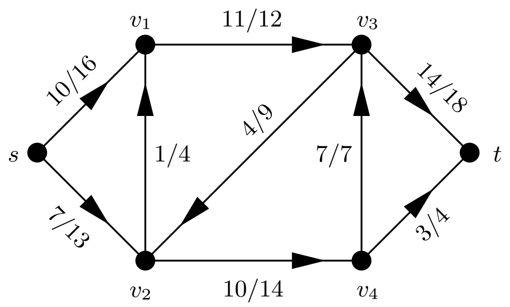
Institut für Theoretische Informatik
Emo Welzl, Jiří Matoušek

26.05.2005

Theoretische Informatik (Kernfach) *SS 2005*
Exercise Set 7

Exercise 1

Consider the following network (G, s, t, c) together with the given initial flow f_0:



Simulate one phase of Dinits algorithm on this network (let f_0 be the current flow at the beginning of this phase). In particular write down the residual network and the graph H (initially $H = \mathrm{SPE}(G_{f_0})$) in every step of the phase. *Hint:* See the backside of this paper.

Exercise 2

Let Q_n be a directed graph of the n-dimensional cube; that is, the vertices of Q_n are represented by zero-one sequences of length n and there is a directed edge from a vertex a to a vertex b if and only if the sequences a and b differ in exactly one coordinate and the value of a in this coordinate is 0 (and 1 in the sequence b).

Choose a source $s = (0, 0, \ldots, 0)$ and a sink $t = (1, 1, \ldots, 1)$. Consider the capacities of all edges to be 1. Find a maximum flow in Q_n, and a maximum flow that is nonzero on all edges of Q_n. If you cannot solve the general case, do at least the case $n = 3$ (make a picture of Q_3).

Exercise 3

Show: For every network with m edges there is a sequence of at most m augmentations which produces a maximum flow.

Exercise 4

Let (G, s, t, c) be a network with all capacities equal to 1. Assume that there is no s-t-cut with capacity k or smaller ($k > 1$ integer). Prove that there are directed s-t-paths P_1, P_2,..., P_k such that P_i and P_j have no common edges for $1 \leq i < j \leq k$. (Two paths may have common vertices.)

Hint: Consider a suitable maximum flow and extract the paths by induction.

Exercise 5

Prove Proposition 3.7.5 in the lecture notes from Hall's theorem.

Due-date: 02.06.05 in the lecture

```
PhaseOfDinits(G, s, t, c, f_0)
  compute the residual network G_{f_0}
  compute SPE(G_{f_0}) = all edges on shortest s-t-paths,
      by two breadth-first searches (from s and t)
  V(H) ← V(G), E(H) ←SPE(G_{f_0}), f ← f_0
  repeat
      P ←SearchPathWithDeletions(H, s, t)
      if P found then
          augment f along P
          delete from H the edges of P that
              leave the residual network
      end if
  until no P found
  return f
```

```
SearchPathWithDeletions(H, s, t)
  v ← s
  P ← the path from s to s with no edges
  while v ≠ t do
      if there is no edge (v, w) ∈ E(H) then
          if v = s then
              return "no path found"
          else
              (u, v) ← be the last edge of P
              delete (u, v) from H
              v ← u
          end if
      else
          (v, w) ← some outgoing edge from v
          extend P by (v, w)
          v ← w
      end if
  end while
  return P
```