Exercise 18.3. Suppose you had an (s,t)-flow f. We know that there exists an (s,t)-path packing of the same size as f; here we are interested in algorithms that take f and compute such a path packing. Such a path packing is called a *flow decomposition* of f.

Design and analyze an algorithm that, in $O(m^2)$ time, computes a maximum path packing x of the same size as f, such that:

1. There are at most m distinct paths (with nonzero value) in x.
2. If f is integral, then x is also integral.

Solution. □

Exercise 18.4. This exercise develops a $O((m^2 + mn\log(n))\log(\lambda))$-time algorithm for maximum (s,t)-flow and builds on ideas from exercise 18.3.

Exercise 18.4.1. Prove the following: Given any (s,t)-flow problem with max flow value $\lambda > 0$, there exists an (s,t)-path where the minimum capacity edge is at least λ/m.

Solution. □

Exercise 18.4.2. Describe an $O(m + n\log(n))$-time algorithm to find the path described above.[4]

[4] $O(m \log n)$ time is a little easier and this running time would still get partial credit. Even if the $O(m+nlog(n))$-running time eludes you, you can assume it as a black box for the next part.

Solution. □

Exercise 18.4.3. Prove the following: Given any (s,t)-flow problem with max flow value $\lambda > 0$, there exists an (s,t)-path where the minimum capacity edge is at least λ/m.

Solution. □