performs an ordering of the urgency-of-service requirements of the programs in the system, finding a "best" (most-urgent requirements) and a "worst" (least-urgent requirements) program. The ordering is determined according to the following rules.

1. Programs in wanted status are better than programs in runnable status; programs in runnable status are better than programs in hung status.

2. Programs which are runnable are ordered according to their queue: queue 1 is best; queue 12 is worst.

3. Programs within a given queue are ordered according to the number of time quanta that they have already received in that queue: the more time a program has already received, the better it is. This rule approximates a "first-in first-out" philosophy, but allows for the fact that several programs within a single queue may each have received some computing time in that queue. (As an example, suppose that program A, in queue 5, computes for two time quanta and then becomes hung. Program B, also in queue 5, then computes for three time quanta and becomes hung. An ordering at this time would make program B better than program A.)

The job of the Swapper can now be defined as finding the best accessible program, and if this program is *better* than the worst program in core, swapping them. As used here the word *swap* means reading one program into memory from a drum storage device and writing another program from the same area of memory onto that device. These two operations may be performed simultaneously when the swapping drum is used. The term *one-direction swap* is defined to mean either one of these operations, but not both. If