Unless otherwise stated, whenever you are asked to compute the approximation ratio of an algorithm you must compute a constant approximation ratio. Please report the best approximation ratio that you can find.

1. A group of $k$ workers in a factory needs to perform a set $T = \{T_1, T_2, \ldots, T_n\}$ of tasks. Each task $T_i$ has to be performed by one worker, it must be started at time $s_i$ and it requires time $p_i$ to be completed (so the task must be finished at time $s_i + p_i$). A worker cannot work on two tasks at the same time, but when a worker finishes a task she can work on another one. The time needed for a worker to move from task $T_i$ to task $T_j$ is $D_{ij}$; so if a worked completes task $T_i$ at time $s_i + p_i$ she cannot perform task $T_j$ if $s_j < s_i + p_i + D_{ij}$. We wish to determine whether the $k$ workers can perform all the tasks in $T$.

   • (15 marks) Design a polynomial time algorithm for solving this problem. The algorithm must return true if all the tasks can be completed by $k$ workers, and false otherwise.

2. Consider the following approximation algorithm for the bin packing problem.

   Algorithm LastFit($I, S$)
   Input: Set $I$ of items and set $S$ of item sizes; item $I_j \in I$ has size $S_j$
   Output: A packing of $I$ into unit size bins
   $B \leftarrow \{\}$
   for each item $I_j \in I$ do {
     if $I_j$ fits in one of the bins of $B$ then
       Put $I_j$ in the last bin where it fits
     else {
       Add a new bin $b$ to $B$
       Put $I_j$ in $b$
     }
   }
   output $B$

   • (10 points) Compute the approximation ratio of the above algorithm. You must explain how you computed the approximation ratio.

3. A set $F = \{f_1, f_2, \ldots, f_n\}$ of files with integer sizes $s_1, s_2, \ldots, s_n$ needs to be stored in a hard disk of capacity $K$. We wish to find a subset of files of maximum total size but not larger than $K$ to be stored in the disk. For example, if we have 4 files with sizes 3, 5, 8, and 6 and $K = 15$, an optimum solution is to store in the hard drive the files of size 6 and 8. Another optimum solution stores the files with sizes 3, 5, and 6. The above problem is NP-hard.

   Consider the following algorithm for the problem.
Algorithm files\( (F, S, n, K) \)

**In:** Set \( F \) of files, set \( S = \{s_1, s_2, \ldots, s_n\} \) of \( n \) file sizes, and hard disk capacity \( K \)

**Out:** Set of files of total size at most \( K \)

\[
\begin{align*}
A & \leftarrow \{\} \\
\text{total} & \leftarrow 0 \\
\text{for } i & \leftarrow 1 \text{ to } n \text{ do} \\
& \quad \text{if } \text{total} + s_i \leq K \text{ then } \{ \\
& \quad \quad \text{Add file } f_i \text{ to } A \\
& \quad \quad \text{total} \leftarrow \text{total} + s_i \\
& \quad \} \\
\text{output } A
\end{align*}
\]

(i) (5 marks) The value \( SOL \) of the solution computed by the algorithm is equal to the total size of the files in set \( A \), i.e. \( SOL = \text{total} \). Show that the approximation ratio, \( OPT/SOL \), of this algorithm is arbitrarily large by giving an instance in which the total size of the files in the set \( A \) returned by this algorithm is very small compared to the value of an optimum solution. Note that the files are not sorted in any particular manner.

(ii) (20 marks) Assume now that the files are sorted in non-increasing order of size. Compute the approximation ratio \( OPT/SOL \) of the algorithm.