Lista de soluções 3

Questão 1

O seguinte algoritmo resolve o problema

1 Calcula $S' = S \mod p = \{s \mod p \mid s \in S\}$
2 Define um polinômio $P(x) = \sum_{0 \le i < p} [i \in S'] x^i$
3 Calcula $Q(x) = \sum_{0 \le i < 2p-1} q_i x^i = P(x)^2$
4 **return** $C = \{i \mod p \mid q_i \neq 0\}$

(Para uma proposiçãp p, $[p] = 1$ caso p é verdadeiro e $[p] = 0$ caso contrário.) O algoritmo funciona porque o coeficiente de x^k em $Q(x)$ satisfaz

$$[x^k]Q(x) = \sum_{\substack{0 \le i < p \\ 0 \le j < p \\ i+j=k}} [i \in S'][j \in S']$$

i.e. o coeficiente é igual ao número de vezes que a soma k pode ser obtida por somar dois números em S'. Logo caso $[x^k]Q(x) > 0$, o $k \in S' + S'$. O passo 1 precisa tempo $O(n)$, os passos 2 e 4 tempo $O(p)$, e o passo 3 tempo $O(p \log p)$ usando a transformada rápida de Fourier para multiplicar dois polinômios. Logo temos tempo $O(n + p \log p)$.

Mais concretamente, o exemplo da lista implementado em GNU R:

```
n=19
C=c(14,15,92,65,35)

Cp=unique(C %% n)
P=numeric(n); P[Cp+1]=1
Pt=fft(P)
Qt=Pt*Pt
Q=fft(Qt,inv=T)/length(Qt)
which(as.double(round(Q))>0)-1
```

Questão 2

Pela questão sabemos que ordenar as tarefas em ordem não-crescente da razão de Smith w_j/p_j é a solução ótima para uma máquina. Seja $C(T)$ o valor da solução ótima para um conjunto de tarefas $S \subseteq [n]$. O problema então é atribuir as tarefas às máquinas. Caso a máquina 1 recebe as tarefas M_1 e máquina 2 recebe $M_2 = [n] \setminus M_1$ e a solução ótima é $C(M_1) + C(M_2)$. Supondo que as tarefas são ordenadas de acordo com w_i/p_i, temos

$$P(j, S_1, S_2) = \begin{cases} \min\{P(j+1, S_1 + p_j, S_2) + w_j(S_1 + p_j), P(j+1, S_1, S_2 + p_j) + w_j(S_2 + p_j)\} & \text{caso } j \le n \\ 0 & \text{caso contrário} \end{cases}.$$

e a solução ótima é $P(1, 0, 0)$. O segundo é terceiro argumento satisfazem $0 \le S_1, S_2 \le P = \sum_{j \in [n]} p_j$, logo o algoritmo precisa tempo e espaço $O(nP^2)$.

Contra-exemplo para o algoritmo guloso O exemplo

j	1	2	3
p_j	1	1	2
w_j	1	1	$2 - \epsilon$

mostra que um algoritmo guloso que processa as tarefas em ordem não-crescente da razão de Smith w_j/p_j e aloca a próxima tarefa na máquina que gera menos custos não funciona: ele aloca as duas primeiras tarefas para as duas máquinas, e depois a terceira para uma das duas. Logo $C_1 = C_2 = 1$, $C_3 = 3$ e o custo total é $8 - 3\epsilon$, enquanto a solução ótima aloca as duas primeiras tarefas para uma máquina e a terceira para uma outra, i.e. $C_1 = 1$, $C_2 = 2$, $C_3 = 2$ com custo total $7 - 2\epsilon$.

Questão 3

Um subproblema adequado é definido por um subjunto das tarefas $S \subseteq [n]$. Temos que sequenciar as tarefas em S a partir do tempo $t(S) = \sum_{j \in [n] \setminus S} p_j$, porque as outras tarefas já foram sequenciadas, e a solução ótima não possui tempo ocioso. Entre todas tarefas em S temos que escolher uma tarefa que minimiza o custo das restantes tarefas. Isso leva à recorrência

$$C(S) = \begin{cases} \min_{j \in S} w_j(t(S) + p_j) + C(S \setminus \{j\}), & \text{caso } S \neq \emptyset, \\ 0, & \text{caso contrário,} \end{cases}$$

com solução ótima $C([n])$; a solução correspondente com PD possui complexidade de espaço $O(2^n)$ e de tempo $O(n2^n)$ porque o cálculo de $t(S)$ e a minimização precisam tempo $O(n)$.