Proposta de solució al problema 1

```
void write_perm_max(int n, int k, int s, vector<int>& sol, vector<bool>& used) {
  if (sol.size() == n) write_solucio(sol);
  else {
    for (int i = 1; i <= n; ++i)
      if (not used[i] and sum_last_k(sol,k-1) + i <= s) {
        used[i] = true;
        sol.push_back(i);
        write_perm_max(n,k,s,sol,used);
        sol.pop_back();
        used[i] = false;
      }
  }
}
```

Proposta de solució al problema 2

(a) Una solució és:

```
int max_valor (const vector<int>& v, int k) {
  if (k < 0) return 0;
  else if (k == 0) return v[0];
  else return max( max_valor(v, k-1), max_valor(v, k-2) + v[k]);
}
```

(b) La recurrència demanada és:

$$T(k) = \begin{cases} 1 & \text{si } k \leq 0 \\ T(k-1) + T(k-2) & \text{si } k > 0 \end{cases}$$

podem veure que el cost de max_valor és major o igual que $T(k)$. A més, tenim que $T(k) = F_{k+1}$. Per tant, el cost del programa principal és major o igual que $T(n-1) = F_n = \Theta(\phi^n)$.

(c) Una solució és:

```
int max_valor(const vector<int>& v) {
  int n = v.size();
  int pre = 0;
  int cur = v[0];
  for (int k = 2; k <= n; ++k) {
    int prepre = pre;
    pre = cur;
    cur = max(pre, prepre + v[k - 1]);
  }
  return cur;
}
```

Si n és la mida del vector v, el cost en espai és $\Theta(1)$, i el cost en temps és $\Theta(n)$.

Les solucions amb cost en espai és $\Theta(n)$, i el cost en temps $\Theta(n)$ no obtindran la puntuació màxima.

Proposta de solució al problema 3

(a) Considerem els intervals $I = \{[0,2],[1,4],[1,4],[3,6],[3,6],[5,7]\}$. L'algorisme primer escollirà el nombre 3, que pertany a 4 intervals. Despres d'eliminar els intervals que contenen 3, només queden $[0,2]$ i $[5,7]$. Escull el nombre 0, que pertany a un interval, i a continuació el nombre 5. En total, es faran tres xerrades.

La solució òptima és fer 2 xerrades, per exemple en els instants 2 i 6.

(b) Primer veurem que donats t_i i t_{i+1}, necessàriament $t_i \leq t_{i+1}$. Això és perquè si $t_i > t_{i+1}$, en la i-èsima iteració l'algorisme hagués escollit t_{i+1} enlloc de t_i. Per veure que la relació ha de ser estricta, ens fixem que en escollir t_i eliminem tots els intervals que contenen t_i. Així doncs, no pot ser que en la següent iteració escollim un $t_{i+1} = t_i$, amb t_{i+1} un extrem d'un interval de la forma $[s, t_{i+1}]$, ja que haguéssim eliminat aquest interval en la iteració i-èsima.

Veure que $\theta \leq g$ és immediat, ja que el nombre de xerrades de la solució òptima, θ, sempre ha de ser menor o igual que el nombre de xerrades de qualsevol altra solució.

L'última propietat la demostrarem per inducció sobre i:

Cas base ($i = 1$): Sabem que t_1 es correspon amb l'extrem superior d'un interval $[s, t_1] \in I$. Ara, per reducció a l'absurd, si $t_1 < t_1^*$, veiem que cap xerrada de la solució òptima està dins $[s, t_1]$, ja que $t_k^* \geq t_1^* > t_1$ per a tota $1 \leq k \leq \theta$, el que contradiu que la solució òptima ha de tenir una xerrada dins cada interval.

Pas d'inducció: assumim que $t_i \geq t_i^*$ i anem a demostrar que $t_{i+1} \geq t_{i+1}^*$. Sabem que t_{i+1} és l'extrem superior d'un interval $[s, t_{i+1}] \in I$. Sabem també que $s > t_i$ ja que si no fos així l'interval $[s, t_{i+1}]$ hauria estat eliminat en la i-ésima iteració. Per tant, tenim $t_i^* \leq t_i < s \leq t_{i+1}$. Per reducció a l'absurd, si $t_{i+1} < t_{i+1}^*$ la cadena exterior es pot allargar fins tenir $t_i^* \leq t_i < s \leq t_{i+1} < t_{i+1}^*$. En aquesta situació, no hi ha cap xerrada t_k^* de la solució òptima que estigui dins l'interval $[s, t_{i+1}]$, el que contradiu el fet que la solució òptima ha de tenir una xerrada dins cada interval.

(c) Assumim que l'algorisme golafre no calcula una solució òptima. És a dir, $g > \theta$. Aleshores l'algorisme golafre planifica la xerrada $(\theta + 1)$-èsima en el temps $t_{\theta+1}$, que és l'extrem superior d'un interval $[s, t_{\theta+1}] \in I$. Aquest interval no pot contenir t_θ ja que altrament hauria estat eliminat. Com que $t_\theta < t_{\theta+1}$, necessàriament $t_\theta < s$. Així doncs, tenim $t_\theta^* \leq t_\theta < s$, i per tant l'interval $[s, t_{\theta+1}]$ no conté cap t_k^*, ja que $t_k^* \leq t_\theta^* < s$, per a tot $1 \leq k \leq \theta$. Això contradiu el fet que la solució òptima ha de tenir una xerrada dins cada interval.

Proposta de solució al problema 4

(a) Considerem $I = \{[1,5],[4,7],[6,10]\}$. L'algorisme només acceptarà la segona reserva, mentre que la solució òptima accepta la primera i la tercera.

(b) Observem primer de tot que per a tot $J \in B_i$ es compleix $|J| \geq |J_i|$, on $|J|$ indica la longitud de l'interval J. Anem a veure que qualsevol solució, i en concret l'òptima, pot escollir com a molt dos intervals dins cada B_i. Si la solució conté $J_i = [a,b]$, aleshores no pot contenir cap altre interval de B_i, ja que tots se solapen amb J_i. Si la solució no conté J_i, raonem per absurd i considerem que la solució escull tres intervals K_1, K_2, K_3. Aquests intervals no poden solapar-se i podem assumir que estan ordenats $K_1 < K_2 < K_3$ i notem $K_i = [a_i, b_i]$. Com que $K_1 \cap J_i \neq \emptyset$ necessàriament $a \leq b_1$. Anàlogament, com que $K_3 \cap J_i \neq \emptyset$, necessàriament $a_3 \leq b$. Per tant, com que els K_i no se solapen tenim $a_2 > b1 \geq a$ i $b_2 < a_3 \leq b$, el que implica que K_2 està inclòs estrictament dins J_i, i això no pot ser ja que J_i és més curt que K_2.

Per acabar, notem que els B_i formen una partició de I. Així doncs, qualsevol interval de la solució òptima pertany a algun B_i, i com que com a màxim en poc escollir dos dins cada B_i i en tenim s, la solució òptima compleix $s^* \leq 2s$, és a dir, $s \geq \frac{1}{2}s^*$.

(c) Existeix un algorisme golafre polinòmic que calcula la solució òptima: inicialment tots els intervals de I estan disponibles. Mentre hi hagi intervals disponibles, repeteix: escull $J = [s,t]$ amb menor t dins els intervals disponibles i marca tots els intervals encara disponibles que se solapen amb J com a no disponibles.