Vito's Family

The famous gangster Vito Deadstone is moving to New York. He has a very big family there, all of them living on Lamafia Avenue. Since he will visit all his relatives very often, he wants to find a house close to them. Indeed, Vito wants to minimize the total distance to all of his relatives and has blackmailed you to write a program that solves his problem.

Input For each test case you will be given the integer number of relatives r ($0 < r < 500$) and the street numbers (also integers) $s_1, s_2, ..., s_i, ..., s_r$ where they live ($0 < s_i < 30000$). Note that several relatives might live at the same street number.

Output For each test case, your program must write the minimal sum of distances from the optimal Vito's house to each one of his relatives. The distance between two street numbers s_i and s_j is $d_{i,j} = |s_i - s_j|$.

Zaino

Scrivere una soluzione dell'algoritmo dello zaino basato su backtrack, in cui vengono elencate tutte le possibili scelte di oggetti che soddisfano i limiti della capienza.

Cubi

Siano dati n cubi. Su ogni faccia del cubo è presente una lettera del'alfabeto. Ogni cubo può essere descritto da una stringa di 6 caratteri (e.g. "ABCDEF"), che rappresenta le 6 facce del cubo.
Sia data una parola costituita da $t \leq n$ caratteri; il vostro compito è descrivere un algoritmo che sia in grado di dire se è possibile comporre la parola utilizzando t degli n cubi, scegliendo *una* faccia per ognuno di essi. Calcolare la complessità dell'algoritmo risultante.

Ad esempio, supponete di avere i seguenti cubi: "ABCDEF", "GHIJKL", "AABBCC", "ISTUVW" e di voler comporre la parola IDEA.

Il gioco delle coppie

Scrivere un algoritmo che, dato un vettore A di n interi distinti (n pari), ritorna **true** se è possibile partizionare A in coppie di elementi che hanno tutte la stessa somma (intesa come la somma degli elementi della coppia), **false** altrimenti. Ad esempio:

$$7, 4, 5, 2, 3, 6$$

può essere partizionato in $7 + 2 = 4 + 5 = 3 + 6$.
Discutere la complessità e la correttezza – per questo esercizio, la dimostrazione di correttezza è importante e va scritta bene.

Torri di controllo

Si consideri un insieme di aerei $A = \{a_1, \ldots, a_n\}$ e un insieme di torri di controllo $T = \{t_1, \ldots, t_m\}$.

In ogni istante, ogni aereo e ogni torre è dotato di coordinate geografiche $(a_i.x, a_i.y)$ o $/t_j.x, t_j.y)$. Ogni torre può gestire al più L aerei, e ovviamente questi devono essere a portata radio r dalla torre.

Scrivere un algoritmo che assegni ogni aereo ad una torre, rispettando i vincoli sulla distanza e sul carico.

Discutere correttezza e complessità.

MC Donald

Gestire un McDonald non è semplice.

- La giornata lavorativa è suddivisa in 3 turni da 4 ore, dalle 11 alle 23.
- Durante il turno $t_i \in T$ (dove T l'insieme di 21 turni), è necessario che siano presenti p_i unità di personale
- Avete a disposizione un insieme D di dipendenti; ogni dipendente $d_j \in D$ dichiara un insieme di turni $n_j \subseteq T$ in cui non può lavorare.
- Ad esempio, d_j non può lavorare nei turni $\{t_1, t_7, t_{21}\}$.
- Per contratto aziendale, ogni lavoratore non può lavorare per più di 5 turni.
- Ogni giorno, un dipendendente non può lavorare per più di due turni (qualsiasi, anche non consecutivi).

Progettare un algoritmo che produca uno scheduling che illustri, per ogni turno, il personale associato e discuterne la complessità.

Spoiler alert!

Vito

Il numero civico ottimale è la mediana.

Si consideri il caso di n dispari e quindi di una singola mediana m; m ha una quantità $(n-1)/2$ di numeri civici sia alla destra che alla sinistra. Il caso con n pari e quindi due valori mediani è simile.

Si consideri ogni altra soluzione m' diversa dalla mediana e assumiamo che $m < m'$ (il caso $m > m'$ è simmetrico).

Sia $d = m' - m$ la differenza fra questi due numeri civici. Nella soluzione in cui abbiamo scelto m', tutti i civici che si trovano a sinistra di m' costano d unità in più rispetto alla soluzione in cui abbiamo scelto m. Tutti i civici a destra di m' (m' incluso) costano d unità in meno rispetto alla soluzione in cui abbiamo scelto m.

Poiché i numeri civici a sinistra di m' sono di più dei numeri civici a destra di m' (m' incluso), la soluzione m' costa più della soluzione m.

Complessità: $O(n)$ per il calcolo della mediana

Zaino

knapsackRec(**int**[ ] P, **int**[ ] V, **boolean**[ ] S, **int** n, **int** i, **int** c)

```
if i == 0 then
    processSolution(S, n)
else
    S[i] = false
    knapsackRec(P, V, S, n, i − 1, c)
    if V[i] ≤ c then
        S[i] = true
        knapsackRec(P, V, S, n, i − 1, c − V[i])
```

knapsack(**int**[ ] P, **int**[ ] V, **int** n, **int** C)

knapsackRec(P, V, n, C)

Il gioco delle coppie – 2012/05/03

checkPairs(**int**[ ] A, **int** n)

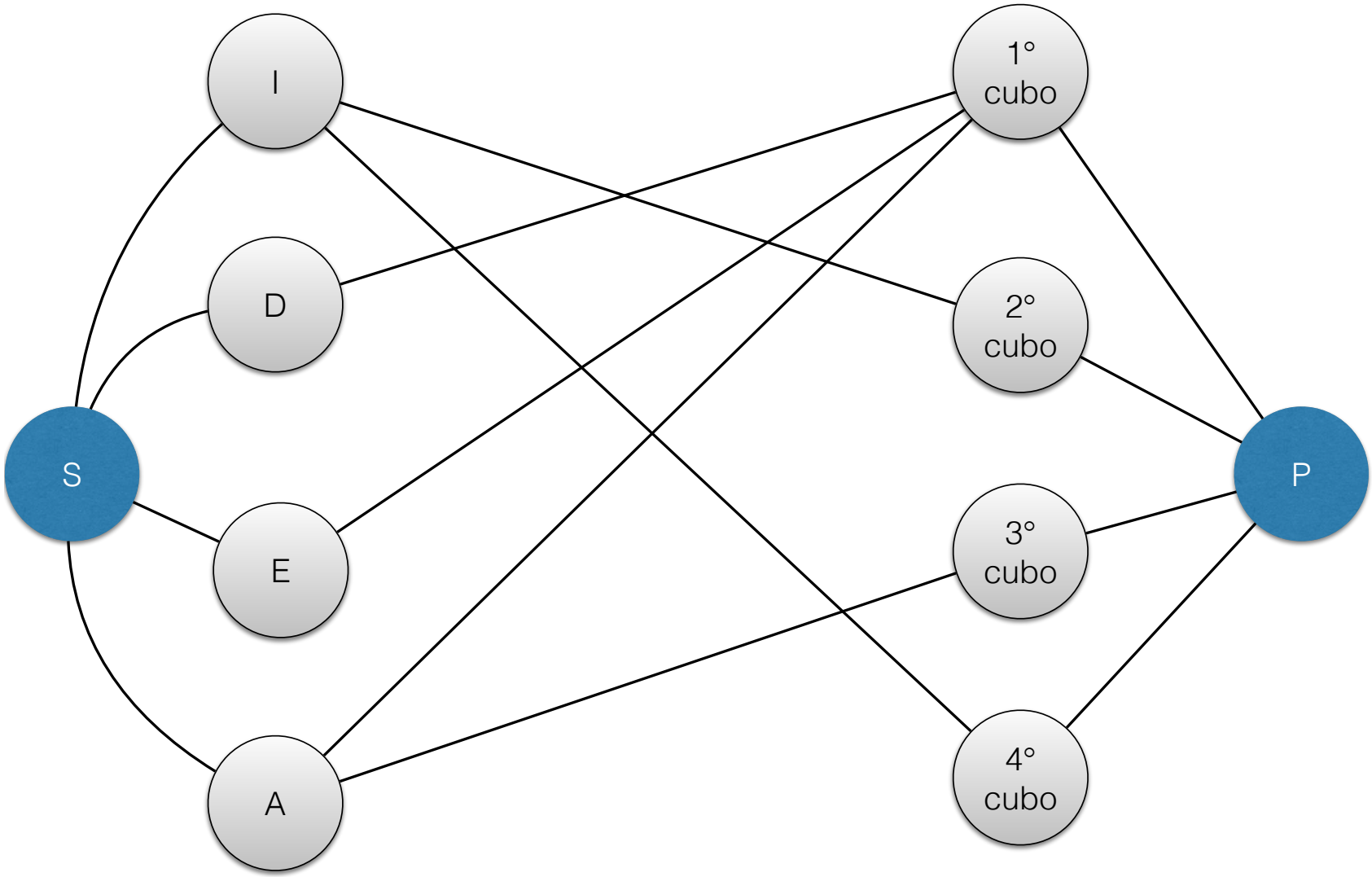
```
sort(A, n)
int pairSum = A[1] + A[n]
for i = 2 to n/2 do
    if A[i] + A[n − i + 1] ≠ pairSum then
        return false
return true
```

Dimostrazione: supponiamo per assurdo che esista un insieme di coppie che rispetti le condizioni per restituire **true**, in cui l'elemento maggiore M sia associato ad un elemento M' diverso dal minore m ($m < M'$). Quindi il minore m è associato ad un elemento m' diverso dal massimo M ($m' < M$). Allora $m + m' < M + M'$, il che contraddice l'ipotesi che tale insieme di coppie rispetti le condizioni per restituire **true**.

Il gioco delle coppie - $O(n)$, hash set

```
checkPairs(int[] A, int n)
int tot = sum(A, n)                    % Sum all elements, O(n)
real pairSum = tot/(n/2)
if pairSum ≠ ⌊pairSum⌋ then
    return false
SET set = Set()                        % Based on a hash table
for i = 1 to n do
    set.insert(A[i])
for i = 1 to n do
    if not set.contains(pairSum − A[i]) then
        return false
return true
```

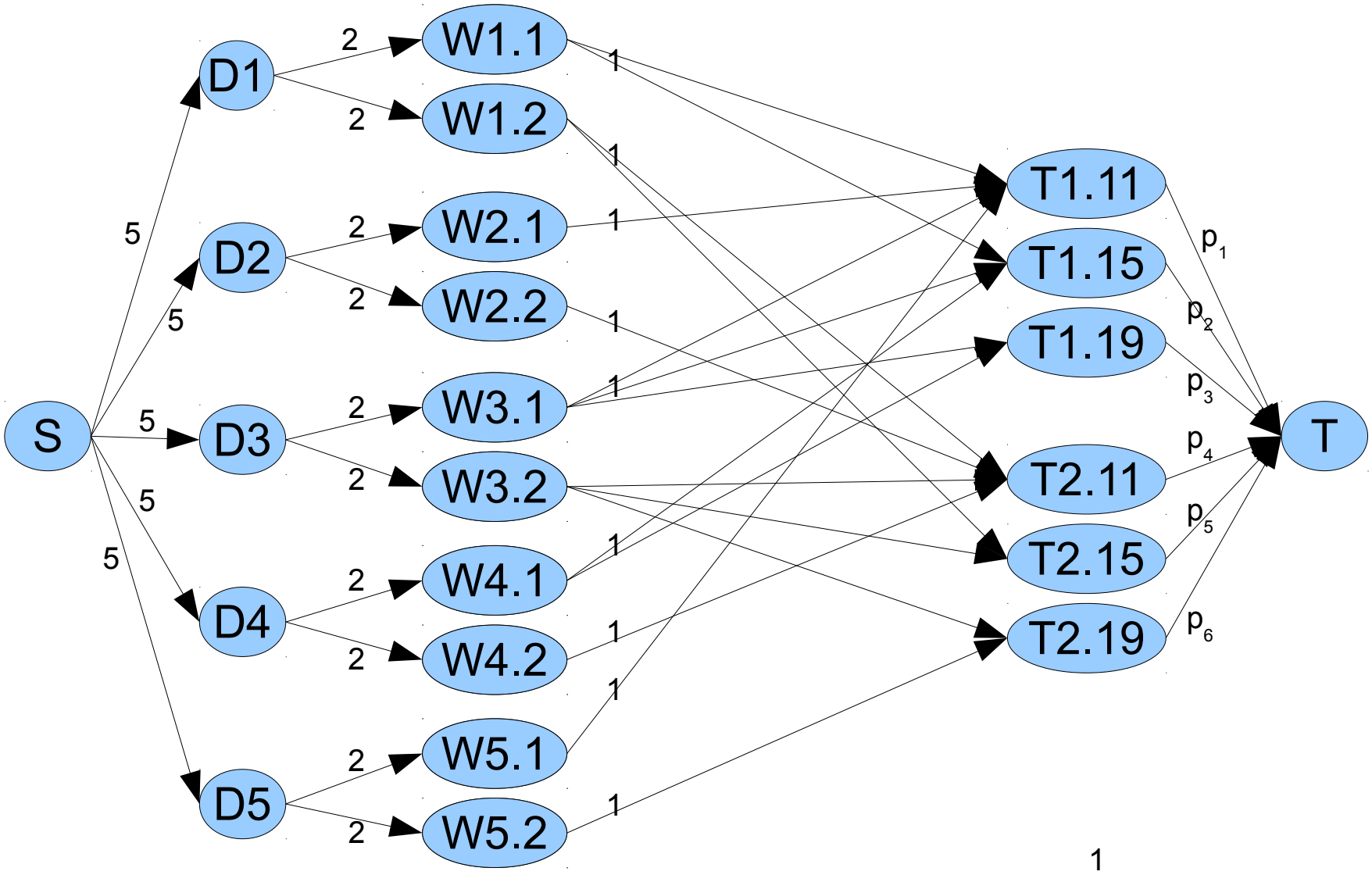
Cubi



Torri di controllo

- È possibile associare ad ogni aereo ed ad ogni torre i vertici di un grafo bipartito.
- Ogni aereo è connesso ad una torre se è entro la sua portata radio con un arco di capacità 1.
- Si aggiunge poi un nodo sorgente s, connesso a tutti gli aerei con archi di capacità 1; e un nodo pozzo p, connesso a tutte le torri con archi di capacità L.
- Il flusso massimo è ovviamente n. Utilizzando il limite derivante da Ford e Fulkerson, il costo dell'algoritmo risultante è quindi $O(n(|V| + |E|))$, dove $|V| = n + m + 2$ e $|E| = O(nm) + n + m$; il costo totale è quindi $O(n^2 m)$.

MCDonald


S
D1
D2
D3
D4
D5
W1.1
W1.2
W2.1
W2.2
W3.1
W3.2
W4.1
W4.2
W5.1
W5.2
T1.11
T1.15
T1.19
T2.11
T2.15
T2.19
T
5
5
5
5
5
2
2
2
2
2
2
2
2
2
2
1
1
1
1
1
1
1
1
1
1
p1
p2
p3
p4
p5
p6
1