Contesteu, sempre justificant la resposta:

Per què l'esquema de backtraking és aplicable per a resoldre aquest enunciat.

és un problema d'optimització, s'ha de trobar el millor repartiment minimitzant les diferencies entre els esquis i cada alumne, cal aplicar l'esquema de trobar la millor solució

Determineu quines decisions ha de prendre el programa.

Per cada alumne quin parell d'esquís li assignem

Completable:

Serà completable mentre quedin alumnes sense esquis i esquís sense assignar.

Solució:

Quan tots els alumnes tinguin un parell assignat

No sempre trobarem solució, si a l'alumne no li va bé cap parell d'esquís per l'alçada o pel numero de peu.

Arbre de decisions:

L'alçada es **exacte** i correspon al numero d'alumnes

L'amplada es **exacte** i correspon al numero de parells d'esquís, cada alumne amb un parell d'esquís

Acceptable:

Hem de comprovar l'alçada i el numero de peu de l'alumne amb l'equí assignat

Marcatge:

Farem servir el marcatge per indicar les parelles d'esquís que ja estan assignades

Codi de la solució:

https://github.com/santiHerranz/ProgramacioAvancada-20170110-Examen

2. Constructor de la classe Solucio

```
public Solucio(int n) {
    N = n;
    solucio = new Parell[n];
    millor = new Parell[n];
    totsAlumnes = new Alumne[n];
    totsEsquis = new ParellEsqui[n];
    TotesDades(totsAlumnes, n, totsEsquis);
}
```

3. Redefiniu el mètode toString

```
public String toString() {
    String s = "";
    if ( millor[0] != null ) {
        s += "Millor: ";
        for (int i=0; i < millor.length; i++){
            s += "("+ millor[i].a.getNom();
            s += ": "+ millor[i].p.getLlargaria() +" cm )\t" ;
        }
    } else
        s += "No hi ha cap solució";
    return  s;
}
```

4. Implementeu el mètode main

```
public static void main (String[] args) {
    System.out.print("Numero alumnes?");
    int n = Keyboard.readInt();
    if (n>0) {
        Solucio s = new Solucio(n);
        s.backSolucio(0, null);
        System.out.println(s.toString());
    }
}
```

5. Mètode backtraking

```
public void backSolucio(int k, boolean[] marcats) {

    // iniciem marcatge d'esquis si es la primera crida
    if ( marcats == null  ) {
        marcats = new boolean[totsEsquis.length];
        for ( int i = 0; i < marcats.length; i++  )
            marcats[i] = false;
    }

    // Serà solucio si arriba a la fulla
    if ( k == N ) {

        System.out.println("Solució: "+ Arrays.toString(solucio) + " => "+
calculDiferencies(solucio) +" cm");

        // guardem millor solució o si es la primera trobada
        if (calculDiferencies(solucio) <  calculDiferencies(millor) || millor[0] == null)
            for (int m = 0; m < solucio.length; m++)
```

```
                millor[m] = new Parell(solucio[m].a, solucio[m].p);

    } else {
        // repartim els esquies
        for (int i = 0; i < marcats.length; i++ ) {
            // assignem un parell d'esquis que no estiguin assignats
            if ( !marcats[i] ) {
                // marquem el parell assignat
                marcats[i] = true;

                Parell p = new Parell(totsAlumnes[k],totsEsquis[i]);

                // comprobem si el parell d'esqui va bé a l'alumne
                if (acceptable(p)) {
                    solucio[k] = p;

                    // mentre sigui completable cridem recursivament
                    backSolucio(k+1, marcats);
                }
                // desmarquem el parell assignat
                marcats[i] = false;
            }
        }
    }
}
```

```
private boolean acceptable(Parell p) {
    boolean peuCorrecte = p.a.getNumeracioPeu() >= p.p.getNumeracioPeuMinim() &&
p.a.getNumeracioPeu() <= p.p.getNumeracioPeuMaxim();
    boolean alcadaCorrecte = 5 >= Math.abs(p.a.getAlcada()-p.p.getLlargaria());
    return peuCorrecte && alcadaCorrecte;
}

private int calculDiferencies(Parell[] parells) {
    int result = 0;
    for (Parell p: parells) {
        if (p != null)
            result += Math.abs(p.a.getAlcada() - p.p.getLlargaria());
    }
    return result;
}
```