Divide-and-Conquer Algorithms
Application to the Closest Pair Problem

2 Specification and Brute-Force Algorithm

Questions 1, 2, 3

Simple réécriture de l'énoncé.

Question 4

brute_force_search_sub_array – La précondition décrit que $a[low..high-1]$ est un sous-tableau de a de longueur au moins 2, et *closest_pair_post_for* fournit exactement la postcondition voulue. Les invariants décrivent que $(!min, !f, !s)$ est toujours un triplet de réponse autorisé (indices distincts dans les bornes, $!min$ distance entre les points correspondants), et de plus $!min$ est inférieur aux distances pour les couples de points déjà parcourus dans la boucle considérée. Pour la boucle sur i, il s'agit des couples d'indices valides x et y avec au moins un d'eux (sans perdre de généralité x) strictement inférieur à i. Pour celle sur j, les couples d'indices i et y, $i+1 \leq y < j$. Si on s'en tient là (sans le dernier assert avec *m_loop_i*) on n'arrive pas à montrer la conservation de l'invariant de la boucle sur i exprimant que $!min$ est inférieur aux distances déjà étudiées, car pour passer de i à $i+1$ on n'arrive pas à réutiliser le résultat pour i qui porte sur $!min$ qui a pu être modifié au cours de la boucle sur j. Une solution consiste à reparler des couples avec $x < i$ dans la boucle sur j (invariant commenté, qui peut remplacer les deux derniers invariants) car pour une étape de cette boucle, le prouveur se rend compte que la nouvelle valeur de *min* est inférieure ou égale à la précédente. Mais j'avais envie de garder mon invariant ne portant que sur les "nouveaux" couples explorés dans la boucle sur j et de comprendre comment rajouter juste ce qu'il manquait, et cela a fonctionné en ajoutant que $!min$ au cours de la boucle sur j est toujours inférieur à la valeur de *min* à l'entrée de la boucle.

brute_force_search – Le tableau doit être de longueur au moins 2, et *closest_pair_post* fournit exactement la postcondition voulue.

3 Closest Pair in 1D

Question 5

Un invariant exprimant que $(!min, !f, !f + 1)$ est une réponse valide pour le sous-tableau $a[0..i]$ a suffi.