Exercise 9.1 Partitioned Boolean Quadratic Problem (PBQP)

Prove that finding a solution for a PBQP to be NP-hard by reducing SAT to PBQP.

*Hint:* Reconsider the NP-hardness proof for register allocation. First, try to map the boolean formula \((a \land b) \lor \neg b\) from the example in Figure 2 of Koes' paper to PBQP. Then, you can derive an algorithm to map any SAT problem to PBQP. Generally, to map \(a \lor b\) you will need four nodes: one for \(a\), one for \(b\) one for \(\lor\) and an auxiliary node.

Exercise 9.2 PBQP Applied

1. Study the LLVM-IR program below and draw the value graph for the loop body (for.body). Include constants, function arguments and PHI nodes from other blocks in the graph. Furthermore, replace the getelementptr instruction by appropriate scalar operations (add/mul) and fold constant expressions together. Assume the size of an i32 is 4 bytes.

2. Use the patterns on the PBQP slide 19 and the cost shown below to create a PBQP instance *only* for the graph constructed in part 1. Assume the patterns AC and A are also available for multiplications (MC/M).

3. Use the optimality-preserving reductions and the heuristic reduction to find a solution for the PBQP problem. Write down the order edges/nodes are eliminated and the rule that was applied.

```llvm
define i32 @array_sum(i32* %A, i32* %B, i32 %N) {
  entry:
    ; preds = %for.body, %entry
    br label %for.cond
  for.cond:
    ; preds = %for.body, %entry
    %iv = phi i32 [ 0, %entry ], [ %iv.inc, %for.body ]
    %sum = phi i32 [ 0, %entry ], [ %add1, %for.body ]
    %B.cur = phi i32* [ %B, %entry ], [ %B.idx, %for.body ]
    %cmp = icmp slt i32 %iv, %N
    br i1 %cmp, label %for.body, label %for.end
  for.body:
    ; preds = %for.cond
    %A.idx = getelementptr i32, i32* %A, i32 %iv
    %B.idx = getelementptr i32, i32* %B, i32 1
    %A.val = load i32, i32* %A.idx, align 4
    %B.val = load i32, i32* %B.idx, align 4
    %add1 = add i32 %sum, %A.val
    %add2 = add i32 %add1, %B.val
    %iv.inc = add i32 1, %iv
    br label %for.cond
  for.end:
    ; preds = %for.cond
    ret i32 %sum
}
```