組み合わせ最適化入門レポート課題

ソースコードはこのgithubにもある。

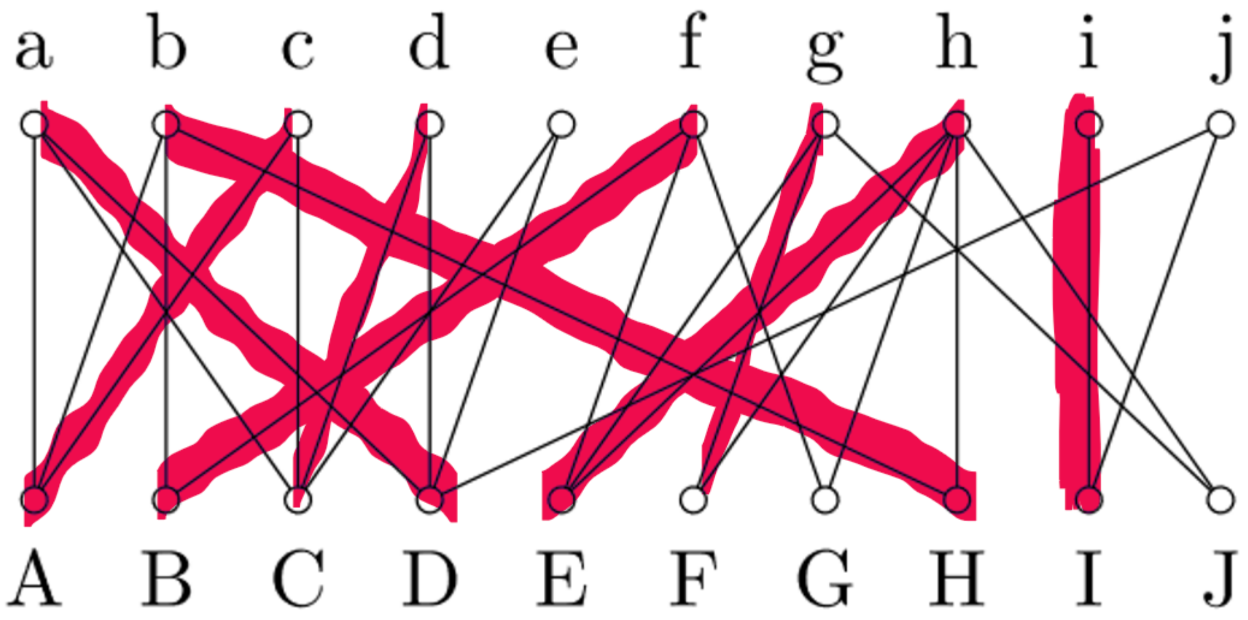
1.最大マッチング

以下のプログラムを用いて、最大マッチングを求めた。

```
#include <algorithm>
#include <iostream>
#include <utility>
#include <vector>

bool loop(int u, std::vector<int> &u_1, std::vector<int> &v_1,
          std::vector<int> path_pre, std::vector<int> &matches,
          const std::vector<std::pair<int, int>> &edges) {
  auto where = edges.begin();
  while ((where = std::find_if(
              where, edges.end(), [&](const std::pair<int, int> e) {
                return e.first == u &&
                       !std::any_of(
                           matches.begin(), matches.end(),
                           [&](const int m) { return edges.at(m) == e; }) &&
                       !std::any_of(path_pre.begin(), path_pre.end(),
                                    [&](const int p) {
                                      return edges.at(p).second == e.second;
                                    });
              })) != edges.end()) {
    auto path = path_pre;
    path.push_back(where - edges.begin());
    auto v_elem = (*where).second;

    auto is_v0 = !any_of(v_1.begin(), v_1.end(),
                         [&](const int v1_elem) { return v1_elem == v_elem; });

    if (is_v0) {
      // u_1,v_1更新
      u_1.push_back(edges.at(path.at(0)).first);
      v_1.push_back(v_elem);

      auto p_size = path.size();

      // matches^pathを消す
      for (std::size_t i = 1; i < p_size; i += 2) {
        matches.erase(std::find(matches.begin(), matches.end(), path.at(i)));
        std::cout << "delete match: " << edges.at(path.at(i)).first << " "
                  << edges.at(path.at(i)).second << std::endl;
      }
      // path/matchesを加える
      for (std::size_t i = 0; i < p_size; i += 2) {
        matches.push_back(path.at(i));
        std::cout << "add match: " << edges.at(path.at(i)).first << " "
                  << edges.at(path.at(i)).second << std::endl;
      }
      return true; //増加道が見つかった
    } else {
      auto itr_match =
          std::find_if(matches.begin(), matches.end(), [&](const int m) {
            return edges.at(m).second == v_elem;
          });

      if (itr_match == matches.end()) {
        continue; //行き止まり
      } else {
        path.push_back(*itr_match);
        if (loop(edges.at(*itr_match).first, u_1, v_1, path, matches, edges)) {
          break;
        };
      }
    }
    where += 1;
  }
  return false; //行き止まり
}

int main() {
  std::vector<std::pair<int, int>> edges;
  edges.push_back({0, 0});
  edges.push_back({0, 2});
  edges.push_back({0, 3});
  edges.push_back({1, 0});
  edges.push_back({1, 1});
  edges.push_back({1, 7});
  edges.push_back({2, 0});
  edges.push_back({2, 2});
  edges.push_back({3, 2});
  edges.push_back({3, 3});
  edges.push_back({4, 2});
  edges.push_back({4, 3});
  edges.push_back({5, 1});
  edges.push_back({5, 6});
  edges.push_back({6, 4});
  edges.push_back({6, 5});
  edges.push_back({6, 9});
  edges.push_back({7, 4});
  edges.push_back({7, 5});
  edges.push_back({7, 7});
  edges.push_back({7, 9});
  edges.push_back({8, 8});
  edges.push_back({9, 3});
  edges.push_back({9, 8});

  std::vector<int> matches = {};
```

```
    std::vector<int> path = {};
    std::vector<int> u_1 = {};
    std::vector<int> v_1 = {};
    std::vector<int> u_all = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9};

    // U_0からV_0へたどり着けるか

    bool is_found = true;

    while (is_found) {
      is_found = false;
      for (auto u : u_all) {
        if (!std::any_of(u_1.begin(), u_1.end(),
                         [&](int u1) { return u == u1; })) {
          if (loop(u, u_1, v_1, path, matches, edges)) {
            is_found = true;
            break;
          }
        }
      }
    }

    std::cout << matches.size() << std::endl;
  }
```

以上のプログラムを実行した結果、以下のようになり、最大マッチングは 8 である。



最大性の証拠は、増加道がなくなったことである。

2 安定マッチング

以下のプログラムを用いた。

```
#include <algorithm>
#include <iostream>
#include <vector>

void loop(std::vector<std::vector<int>>& rank_first, const
std::vector<std::vector<int>>& rank_second, std::vector<int>& result, int
choosing)
{
    if (choosing == rank_first.size()) {
        //loop終了
    } else {
        auto target = rank_first.at(choosing).at(0);
        auto lival = std::find(result.begin(), result.end(), target) -
result.begin();
        if (lival == result.size()) {
            if (choosing == result.size()) {
                result.push_back(target);
            } else {
                result.at(choosing) = target;
            }
            //std::cout << "choosing: " << choosing << " target: " << target << "
no lival" << std::endl;
            loop(rank_first, rank_second, result, result.size());
        } else {
            auto choosing_rank = std::find(rank_second.at(target).begin(),
rank_second.at(target).end(), choosing) - rank_second.at(target).begin();
            auto lival_rank = std::find(rank_second.at(target).begin(),
rank_second.at(target).end(), lival) - rank_second.at(target).begin();
            //std::cout << "choosing: " << choosing << " target: " << target << "
lival: " << lival;
            if (choosing_rank > lival_rank) {
                rank_first.at(choosing).erase(rank_first.at(choosing).begin());
                loop(rank_first, rank_second, result, choosing);
            } else {
                rank_first.at(lival).erase(rank_first.at(lival).begin());
                if (choosing == result.size()) {
                    result.push_back(target);
                } else {
                    result.at(choosing) = target;
                }
                loop(rank_first, rank_second, result, lival);
            }
        }
    }
}


int main()
{
    std::vector<std::vector<int>> rank_first = {{4, 5, 2, 1, 0, 3},
        {0, 2, 5, 4, 1, 3},
        {4, 0, 3, 2, 1, 5},
        {4, 0, 5, 1, 3, 2},
            {0, 2, 1, 5, 4, 3},
            {3, 2, 1, 0, 4, 5}};

    std::vector<std::vector<int>> rank_second = {{2, 5, 3, 1, 4, 0},
            {3, 5, 0, 2, 1, 4},
            {3, 0, 5, 1, 2, 4},
            {5, 4, 1, 0, 3, 2},
            {4, 2, 1, 5, 0, 3},
            {0, 3, 4, 5, 2, 1}};

    std::vector<int> result;

    loop(rank_first, rank_second, result, 0);

    for (auto target : result) {
        std::cout << target << " ";
    }
    std::cout << std::endl;
}
```

	1st	2ed	3rd	4th	5th	6th
U	E	**F**	C	B	A	D
V	A	**C**	F	E	B	D
W	**E**	A	D	C	B	F
X	E	**A**	F	B	D	C
Y	A	C	**B**	F	E	D
Z	**D**	C	B	A	E	F

	1st	2ed	3rd	4th	5th	6th
A	W	Z	**X**	V	Y	U
B	X	Z	U	W	V	**Y**
C	X	U	Z	**V**	W	Y
D	**Z**	Y	V	U	X	W
E	Y	**W**	V	Z	U	X
F	**U**	X	Y	Z	W	V

以上の条件だと、下記の結果になった。

結果 U-F, V-C, W-E, X-A, Y-B, Z-D

優先度(firstとsecond)を入れ替えた結果 A-W, B-X, C-V, D-Z, E-Y, F-U