Универзитет Унион

Рачунарски факултет

Лазар Миленковић

# Поређење похлепних алгоритама и динамичког програмирања за стратегију игре Таблић и њених модификација

10.07.2017

| | |
|---|---|
| Кандидат: | Лазар Миленковић |
| Број индекса: | РН 8/13 |
| Наслов: | Поређење похлепних алгоритама и динамичког програмирања за стратегију игре Таблић и њених модификација |
| Ментор: | Др Владимир Миловановић |

**Апстракт**

У овом раду пореде се временска, меморијска сложеност, као и ефикасност више похлепних стратегија, као и стратегија динамичког програмирања за игру Таблић и њених модификација. Таблић је популарна игра картама у којој сваки играч наизменично повлачи једну карту са циљем да на крају игре има највећи збир сакупљених карата. У стандардној верзији, играч зна само карте које му се тренутно налазе у руци као и све карте које су прошле до сада. За ову верзију игре имплементирано је неколико похлепних стратегија, као и стратегија динамичког програмирања. Модификована верзија подразумева да играч зна којом стратегијом играју његови противници, као и редослед карата до краја партије. За ову верзију су такође имплементиране похлепна стратегија, динамичка стратегија, као и бектрек стратегија која гарантовано налази најбољи низ потеза.

# Садржај

# 1 Увод

Сматра се да су карташке игре откривене у Кини у деветом веку нове ере. До данас су толико распрострањене да представљају средство разоноде широм читавог света. Неке игре попут преферанса, блекџека и покера су толико популарне да постоје професионални турнири са позамашним наградним фондовима. Због такве популарности многи математичари се баве изучавањем карташких игара и њихових особина, покушавајући да нађу победничке стратегије или докажу непостојање истих. З последњих неколико година велики тренд су постали турнири у карташким играма на Интернету, где корисници такође могу зарађивати новац играјући против рачунара или против других играча. Овакав тренд повлачи велики број информатичара који имплементирају ботове за играње тих игара.

Овај рад се бави игром Таблић која је јако популарна на Балкану. У игри обично учествују два играча који користе цео шпил од 52 карте. На почетку се поставе четири карте на сто и сваком играчу се подели по шест карата. Играчи играју наизменично и у сваком потезу могу изабрати скуп карата са стола који у суми дају вредност неке карте из његове руке. Уколико постоји такав скуп, играч ставља све карте из скупа као и одговарајућу карту из руке на своју гомилу. Друга могућност играча је да произвољну карту из руке стави на сто. Када играчима понестане карата, подли им се опет по шест карата. Игра се завршава након четири круга дељења ($4+6\cdot 2\cdot 4 = 52$). Победник је онај играч који има највише штихова [1], а уколико је тај број исти онда је победник онај играч који је сакупио све укупно више карата. Могуће су верзије игре са четири такмичара где су обично два играча у једном тиму, као и верзија са три играча где сваки играч игра за себе. Верзија са три играча подразумева да се у последњем кругу дељења сваки играч добије по четири карте уместо шест. Правила игре су детаљно описана у другом поглављу.

Испитују се особине ботови чије стратегије се заснивају на похлепним, бектрек и генетским алгоритмима. Посматране су успешност сваког од алгоритама, као и временске перформансе. Успешност је мерена симулирањем више партија где су противници два различита бота. Временске перформансе су најпре теоријски испитиване анализом сложености алгоритма, а након тога је мерено време специфичних имплементација. Неки од ботова који су имплементирани имају предност у виду познавања противникових карата или редоследа свих карата у шпилу. Оваква предност је неопходна да би алгоритми као што је бектрек уопште функционисали.

Мотивација за ову тему је једна од лекција курса *Увод у алгоритме, 6.006* са Масачусетског Института за Технологију [2]. У лекцији је описа-

[1] штихови су карте $10, A, J, Q, K$

но оптимално играње Блекџека уз познавање свих карата до краја игре. Алгоритам који је разматран користи динамичко програмирање као парадигму.

Друго поглавље бави се историјом таблића и њеним правилима. Описују се све специфичне ситуације које се могу догодити током партије. Разматра се и неколико стратегија које играчи обично користе. Ово поглавље представља потребан увод за разумевање стратегија које су описане у трећој секцији. Треће поглавље описује сваку од коришћених стратегија, мотивацију за њиховим коришћењем, теоријску анализу успешности и сложености, као и имплементацијске детаље. У поглављу са резултатима описује се начин на који су ботови тестирани и анализира добијене податке.

# 2 О игри Таблић

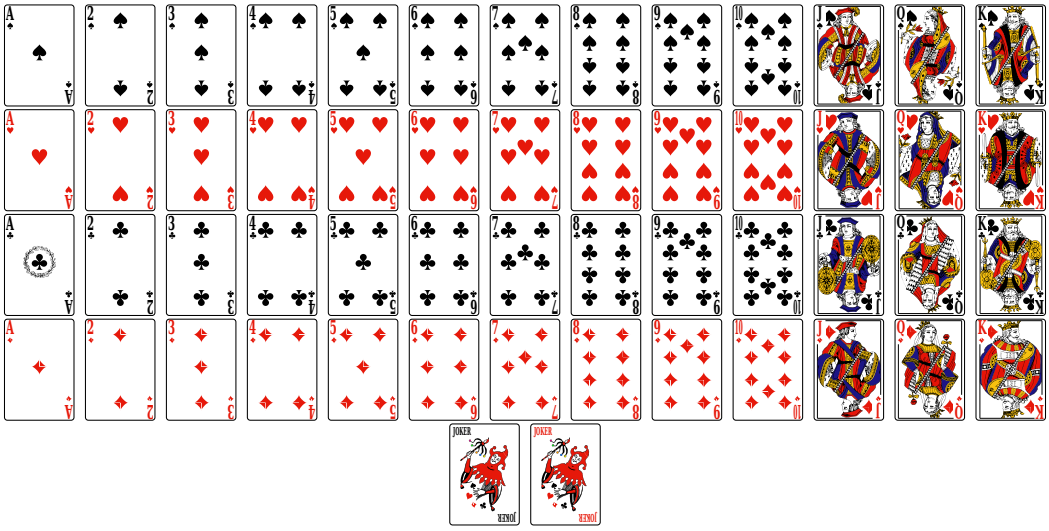
## 2.1 Историја карташких игара

Сматра се да су прве карташке игре настале у 9. веку нове ере у Кини за време владавине династије Танг. Ова претпоставка везана је за текст који описује ћерку тадашњег владара како 868. године игра „игру листова“ са члановима породице свог мужа. Постоји књига о овој игри за коју се претпоставља да ју је написала управо владарева кћи. Карташке игре повезују се открићем технике штампања на папир која је између осталог коришћена и за штампање шпилова карата.

Играње карата се проширило из Кине у Персију и Индију, где су шпилови имали различите боје односно знаке, налик савременим шпиловима. Сваки знак је имао 12 карата поређаних тако да карте са највећим бројевима осликавају краљеве и везире. До 11. века су се карташке игре раширile по целој Азији а дошле су и у Египат. Археолози су нашли египатске карте који потичу из 12. века. Њихови шпилови су имали 52 карте подељене у четири знака. Сваки знак је имао 10 обичних карата и 3 карте највеће вредности које осликавају владаре по хијерархији. Овакав изглед шпила доста личи на модеран шпил који је данас у употреби широм света. Оваква верзија шпила најпре је стигла у јужне делове Европе средином 14. века. Једина промена јесте у знацима који су се користили. У стом веку карташке игре су се прошириле и у Француску, Каталонију и Швајцарску. Ускоро се у Немачкој појављују професионални произвођачи карташких комплета који би штампали шпилове. Те штампане шпилове би касније ручно осликавали уметници.

Први симболи који се данас користе, листови и срца, уведени у Немачкој где су касније уведени и детелина и пик. Највећа карта која је означавала краља појављује се 1377. године у Немачким и Швајцарским шпиловима, где се први пут спорадично појављује и краљица на тринаестој карти. Још један битан моменат у развоју јесте и осликавање броја на ивице карте због лакшег држања у једној руци. Први овакав шпил датира из 1693, док се учестанија употреба усталила тек након 18. века. Могућност ротирања, односно симетрично осликавање датира из 1745. године. Ова карактеристика карата убрзо бива забрањена од Француске владе која је у то време контролисала дизајн. У другим европским земљама се овај дизајн усталио све до модерних шпилова. Заобљене ивице на картама настале су јер се оштре ивице брзо оштете што пружа могућност играчима да отркију о којој карти је реч на основу оштећења. Текстуре на полеђини карте настају из сличних разлога: карте са текстурама прикривају могућа ожтећења полеђине. Први шпилови који су садржали џокере појављују се у Америци где је у то време постојала игра која их користи. Постоји референца из 1875 која помиње употребу џокера као карте која мења другу карту, што је у данашњим играма обично случај [1].

Шпилови који се данас користе састоје се од 52 карте и џокера. Карте су подељене у четири знака: лист (пик), детелина (треф), ромб (каро) и срце (херц). Сваки од ових знакова има по 13 карата $A, 2, 3, 4, 5, 6, 7, 8, 9, 10, J, Q, K$, где $A$ представља аса, $J$ представља жандара, $Q$ краљицу, а $K$ краља.



**Слика 1:** *Модеран шпил карата састоји се од 52 карте у четири различита знака и џокера.*

## 2.2 Неке особине карташких игара

Карташким играма сматра се било која друштвена игра која подразумева коришћење горе описаног шпила. Захваљујући дугогодишњој историји карташког шпила, до данас постоји широк спектар различитих игара. Многе од игара имају светски прихваћена правила, док се у другима правила разликују од културе до културе.

Већина карташких игара има предефинисан број играча који могу истовремено играти једну партију. Разликујемо игре са једним играчем (солитер и пасијанс игре), игре са два играча (таблић обично сврставамо у ову категорију) и игре са више од два играча. Многе игре омогућавају проширење са два на четири играча поделом у партнерске парове. Обично ови парови седе један наспрам другог тако да једни другима не могу видети карте. Партнери могу размењивати информације које се не односе на карте које имају тренутно у руци. Други начин проширења игре јесте да сваки играч игра за себе и овакво проширење је скоро увек могуће и лимитирано је само на број карата у шпилу. Током партије која укључује два играча, оба играча углавном имају у руци само одређени број карата из шпила, јер би у случају поседовања свих карата играчи могли да знају све противникове карте.

У већини игара играчи играју по један потез у унапред одређеном смеру (нпр. редослед казаљке на сату уколико седе за столом). Обично је један играч, *дилер*, одређен да меша шпил и дели карте осталим играчима. У зависности од игре ово може бити предност или мана. Део игре између два дељења назива се *рука*, а тако се назива и скуп карата које један играч има код себе након дељења. Једна игра се састоји од неколико рука и завршава се обично када се потроши комплетан шпил или када неки играч пређе унапред дефинисан број поена. Многе игре захтевају да шпил буде добро промешан због подједнаких вероватноћа победе свих играча. На професионалним турнирима постоје дилери који нису играчи и који су тренирани да праведно мешају карте и вешто их поделе.

Након што дилер измеша карте, наредни играч по редоследу игре *сече* шпил. Сечење је потез у коме се шпил подели на две целине и пребаци се горња испод доње. Током дељења сваки играч добија карте једну по једну или више одједном у редоследу игре. Битно је да током дељења карте остану окренуте лицем ка столу тако да нико не може видети туђе карте. У неким играма се такође одређени број карата ставља на сто лицем окренутим на горе и те карте формирају *талон*. Остатак карата које нису подељене остављају се по страни за наредни круг дељења.

Скоро све игре настале су у малом кругу људи и одатле се ширили у зависности од интересантности. Таблић је игра која је настала на Балкану и углавном није позната у осталим деловима света. Постоје и светски

познате игре које имају интернационална правила и организације које воде рачуна о њима. Игра бриџ има интернационалне турнире које организују Светска бриџ федерација [6], као и локалне организације попут Бриџ савеза Србије [7] или Америчке контракторске бриџ лиге [5]. Још један пример светски познате игре је покер који се јавља са различитим модификацијама широм света. Постоје интернационални турнири са устаљеним правилима као Светски серијал у покеру [8] и Светски покерски турнир [9]. Правила ових организација морају се поштовати само унутар турнира које оне организују. Углавном се правила покера који се игра на осталим местима разликује у неком детаљу.

Већина правила за професионалне турнире има дефинисан скуп понашања играча који се сматра *варањем*. Играчи бивају кажњени у случају да буду ухваћени у варању. Постоје и ситуације где играч случајно види туђу карту или одигра потез када није ред на њега. Ове радње обично захтевају поништавање партије. Сматра се да играч који случајно види туђу карту а не пријави то такође вара.

## 2.3 Типови игара

Карташке игре могу се поделити у породице сличних игара које поседују неке заједничке особине.

*Трик игре.* У сваком потезу играчи извуку једну карту из своје руке и у зависности од вредности извучене карте један играч осваја појене. У зависности од игре победник је онај који освоји највише руку, најмање руку или тачно оређен број руку. Бриџ је пример овакве игре.

*Игре комбиновања.* Циљ оваквих игара је сакупити одређену колекцију карата пре осталих играча. Популарни представник ових игара је реми, где је циљ сакупити низ карата истог знака или све знакове једног броја. Победник је играч који се први ослободи свих карата из руке.

*Игре паковања.* Као што име налаже, циљ оваквих игара је „спаковати“ све карте из руке на сто. Блекџек и уно су популарни представници ове породице.

*Игре сакупљања.* Победник је играч који сакупи цео шпил. Познати представник је игра рат где играчи извлаче по једну карту и играч са већом картом преузме обе карте код себе.

*Игре пецања.* У овим играма играчи траже са талона карту која одговара карти у њиховој руци. Ове игре су најраспрострањеније у Италији где се скопа сматра националном карташком игром и Кини, где постоји доста различитих игара из ове породице.

*Игре поређења.* У овим играма победник је играч који је има највреднију руку. Представници ове групе су покер и блекџек.

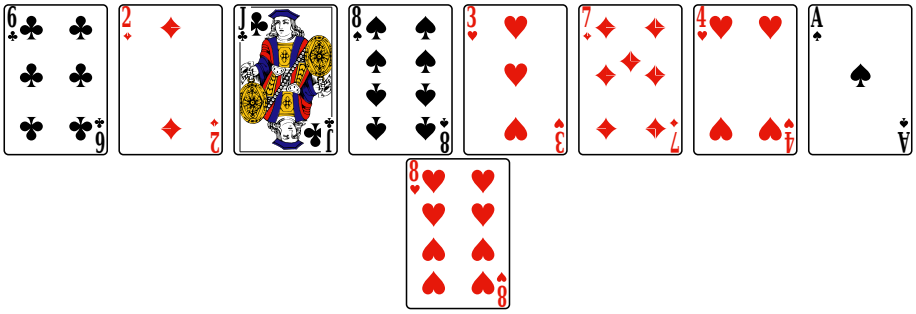
*Пасијенс и солитер игре.* Ове игре намењене су за једног играча који има циљ да на основу задатих правила сложи све карте из руке на талон.

## 2.4 Правила игре таблић

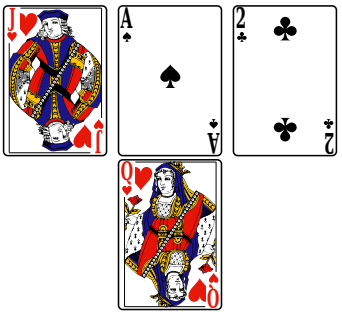
Таблић се најчешће игра у два играча и правила која следе односе се на такву игру [10]. У свакој партији одреди се дилер који дели карте током целе партије. У свакој наредној партији дилери се смењују. На почетку игре, након мешања и сечења шпила, играч који дели остави четири карте на талон. Након што остави карте на талон, дилер подели противнику и себи по једну руку која се састоји од 6 карата. Дилер дели руку тако што наизменично даје по три карте, најпре противнику па себи. Једна рука састоји се од по 6 потеза сваког играча, где потезе играчи повлаче наизменично. Обично први игра играч који није дилер. Игра се завршава када се истроши цео шпил, односно након четири руке.

У једном потезу играч изабере карту из своје руке и покушава да нађе један или више скупова карата са талона који у збиру дају његову карту. То значи да играч може узети све карте са талона које имају вредност као и његова одабрана карта из руке. Поред тога, играч може узети и све комбинације карата са талона које у збиру дају његову извучену карту. Ове комбинације морају бити дисјунктне, односно никоје две комбинације не смеју имати заједничку карту. Вредности карата одређују се њиховим бројем, са додатком да жандар има вредност 12, краљица 13, а краљ 14. Ас карта може у сваком потезу произвољно имати вредности 1 или 11 у зависности од играчеве одлуке. Може се десити да играч не може наћи одговарајуће карте на талону и у том случају треба да изабере произвољну карту из руке и одложи је на талон. Након одиграног потеза играч све сакупљене карте (рачунајући и карту из руке) ставља на своју гомилу. Када се заврши цела игра, преостале карте са талона сакупља играч који је последњи нешто носио са талона.

Победник је играч који на крају сакупи највећи број поена. Најзначајнији фактор за број поена јесте број освојених *штихова*. Штихови су карте $A$, 10, $J$, $Q$ и , где свака од њих носи по један поен уз додатак да десетка каро (дупла десетка) носи два поена. Посебна карта у игри је двојка треф (мала двојка) која такође доноси један поен. Други фактор који утиче на поена је број „очишћених“ табли. Када играч након потеза остави талон празан (очисти талон), добија додатни поен. Уколико је резултат нерешен играч са већим бројем карата на гомили (уколико посотји) добија три додатна поена (три на карте). Победник је играч са већим бројем поена.



**Слика 2:** *Изглед табле представљен је у горњем реду, а корисник жели одиграти 8 срце. На талону се налазе следеће групе које одговарају осмици: шест детелина и два каро, 8 лист, 7 каро и ас лист, док је последња одговарајућа група 3 херц, 4 херц и ас лист. Последње две групе имају заједничког аса тако да се играч мора одлучити за само једну од њих, док су преостале две групе без заједничких чланова тако да их може носити обе. Играч се такође може одлучити и да не носи неку од група, али то обично није оптимална стратегија.*



**Слика 3:** *Играч у руци има краљицу а на табли се налазе жандар, ас и двојка. У овом случају одговарајуће комбинације су жандар и ас (посматрамо га као да има вредност 1), или двојка и ас (посматрамо га као да има вредност 11). Није могуће носити све три карте.*

# 3 Преглед стратегија

## 3.1 Похлепни алгоритам

## 3.2 Бектрек алгоритам

## 3.3 Генетски алгоритам

## 3.4 Остале парадигме

# 4 Резултати

# 5 Закључак

# Додаци

# A Имплементације алгоритама

**Фајл 1:** *Главни програм*

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

#include "utils.h"
#include "backtrack6.h"
#include "backtrack.h"
#include "greedy.h"

using namespace std;

pair<pair<int, int>, pair<int, int>> play_game(vector<int> deck
    ) {
    vector<int> first;
    vector<int> second;
    vector<int> table;
    int curr_player = 0;
    pair<pair<int, int>, pair<int, int>> score = make_pair(
        make_pair(0, 0), make_pair(0, 0));
    for (int i = 0; i < 4; i++) {
        table.push_back(deck.back());
        deck.pop_back();
    }
    int last_player_scored = 0;
    for (int move = 0; move < 48; move++) {
//        cout << first.size() << " " << second.size() << endl;
        cout << "TABLE: ";
        for (int i = 0; i < table.size(); i++) {
            cout << table[i] << " ";
        } cout << endl;
        if (curr_player == 0 && first.empty()) {
            deal(deck, first, second);
        }

        pair<int, int> tmp_score;

        if (curr_player == 0) {
//            tmp_score = greedy_simple(table, first);
            tmp_score = backtrack(table, first, second, deck,
                curr_player == 0);
            score.first.first += tmp_score.first;
            score.first.second += tmp_score.second;
            if (tmp_score > make_pair(0, 0)) {
                last_player_scored = 0;
            }
```

```
        } else {
//            tmp_score = backtrack(table, second, first, deck,
    curr_player == 0);
            tmp_score = greedy_simple(table, second);
            score.second.first += tmp_score.first;
            score.second.second += tmp_score.second;
            if (tmp_score > make_pair(0, 0)) {
                last_player_scored = 1;
            }
        }
        curr_player ^= 1;

    }

    // zadnja karta
    int remaining_valuables = 0;
    for (int i = 0; i < table.size(); i++) {
        if (table[i] >= 10) {
            remaining_valuables++;
        }
    }
    if (last_player_scored == 0) {
        score.first.first += remaining_valuables;
        score.first.second += table.size();
    } else {
        score.second.first += remaining_valuables;
        score.second.second += table.size();
    }
    return score;
}

int main() {
    vector<int> deck;
    for (int i = 2; i <= 14; i++) {
        for (int j = 0; j < 4; j++) {
            deck.push_back(i);
        }
    }
    random_shuffle(deck.begin(), deck.end());
    pair<pair<int, int>, pair<int, int>> score = play_game(deck
        );
    if (score.first > score.second) {
        cout << "First wins!" << endl;
    } else {
        cout << "Second wins!" << endl;
    }
    printf("(%d, %d) - (%d, %d)\n", score.first.first, score.
        first.second, score.second.first, score.second.second);
    return 0;
}
```

**Фајл 2:** *Хедер фајл за бектрек алгоритам*

```
#ifndef TABLIC_BACKTRACK_H
#define TABLIC_BACKTRACK_H

std::pair<int, int> backtrack(std::vector<int> &table,
                              std::vector<int> &hand,
                              std::vector<int> &other,
                              std::vector<int> &deck,
                              bool i_am_the_first_player);

#endif //TABLIC_BACKTRACK_H
```

**Фајл 3:** *Имплементација бектрек алгоритма*

```
#include <vector>
#include <iostream>
#include <map>

using namespace std;

#include "greedy.h"
#include "utils.h"

map<tuple<vector<int>, vector<int>, vector<int>>, tuple<int,
    int, pair<int, int>>> memo6;

pair<int, int> backtrack6_helper(vector<int> &table,
                                 vector<int> &hand,
                                 vector<int> &other,
                                 int depth = 0) {
    if (hand.empty()) {
        return make_pair(0, 0);
    }
    auto tpl = make_tuple(table, hand, other);
    if (memo6.find(tpl) != memo6.end()) {
        auto best_move = memo6[tpl];
        int idx = get<0>(best_move);
        int mask = get<1>(best_move);
        pair<int, int> score = get<2>(best_move);
        do_move(hand[idx], mask, hand, table);
        return score;
    }

    pair<int, int> best_score = make_pair(0, 0);
    pair<int, int> best_curr_score = make_pair(0, 0);
    int best_mask = 0, best_idx = 0;
    int aces = count(table.begin(), table.end(), 11);
    for (int i = 0; i < hand.size(); i++) {
        for (int aces_mask = 0; aces_mask < (1 << aces);
            aces_mask++) {
            // table size can possibly grow big
            for (int mask = 0; mask < (1LL << table.size());
                mask++) {
                int card = hand[i];
                pair<int, int> tmp_score = try_move(card, mask,
                    aces_mask, hand, table);

                vector<int> new_table = table;
                vector<int> new_hand = hand;
                vector<int> new_other = other;
                do_move(card, mask, new_hand, new_table);
```

```
                if (!new_other.empty()) {
                    greedy_simple(new_table, new_other);
                }

                pair<int, int> global_score = backtrack6_helper
                    (new_table, new_hand, new_other, depth + 1);
                global_score.first += tmp_score.first;
                global_score.second += tmp_score.second;
                if (global_score > best_score) {
                    best_score = global_score;
                    best_curr_score = tmp_score;
                    best_idx = i;
                    best_mask = mask;
                }
            }
        }
    }

    do_move(hand[best_idx], best_mask, hand, table);
    memo6[tpl] = make_tuple(best_idx, best_mask,
        best_curr_score);
    return best_curr_score;
}

pair<int, int> backtrack6(vector<int> &table,
                          vector<int> &hand,
                          vector<int> &other) {

    pair<int, int> score = backtrack6_helper(table, hand, other
        );
//    cout << score.first << " " << score.second << endl;
    return score;
}
```

**Фајл 4:** *Хедер фајл за похлепни алгоритам*

```
#ifndef TABLIC_GREEDY_H
#define TABLIC_GREEDY_H

std::pair<int, int> greedy_simple(std::vector<int> &table, std
    ::vector<int> &hand);

#endif //TABLIC_GREEDY_H
```

**Фајл 5:** *Имплементација похлепног алгоритма*

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

#include "utils.h"

using namespace std;

pair<int, int> greedy_simple(vector<int> &table, vector<int> &
    hand) {
    pair<int, int> best_score = make_pair(0, 0);
    int best_mask = 0, best_idx = 0;
    int aces = count(table.begin(), table.end(), 11);
    for (int i = 0; i < hand.size(); i++) {
        for (int aces_mask = 0; aces_mask < (1 << aces);
            aces_mask++) {
            // table size can grow big
            for (int mask = 0; mask < (1LL << table.size());
                mask++) {
                int card = hand[i];
                pair<int, int> tmp_score = try_move(card, mask,
                    aces_mask, hand, table);
                if (tmp_score > best_score) {
                    best_score = tmp_score;
                    best_idx = i;
                    best_mask = mask;
                }
            }
        }
    }
//    cout << "GRDY _OUT " << endl;

    do_move(hand[best_idx], best_mask, hand, table);
//    cout << "GRDY _OUT " << endl;
    return best_score;
}
```

# Литература


[1] http://www.wopc.co.uk/

[2] https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-006-introduction-to-algorithms-fall-2011/recitation-videos/recitation-20-dynamic-programming-blackjack/

[3] https://github.com/emisaacson/Blackjack

[4] Мастер рад неког лика из Холандије

[5] http://www.acbl.org/

[6] http://www.worldbridge.org/

[7] http://www.bridgeserbia.org/

[8] http://www.wsop.com/

[9] http://www.worldpokertour.com/

[10] http://www.igrajkarte.com/blog/post/tablic-pravila/